Class Notes for NACS 643
An Introduction to Computational Neuroscience[1]

Todd Troyer

Draft Manuscript, printed August 26, 2005.

---

[1] Copyright Todd W. Troyer, 2005

Contents

Chapter 1

Introduction

Last changed Jan 26, 2005

1.1 Course Objectives

These notes have three main objectives: (i) to present the major concepts in the field of computational neuroscience, (ii) to present the basic mathematics that underlies these concepts, and (iii) to give the reader some idea of common approaches taken by computational neuroscientists when combining (i) and (ii). Most books on computational neuroscience take one of two approaches. In the first approach, the text is designed for computational students with an interest in neuroscience. A reader must already have significant mathematical knowledge in order to comfortably read the text. In the other approach, the text is designed for a broad audience and the mathematics is separated from the presentation of the biology. The main narrative focuses on the underlying concepts discussed within the field of computational neuroscience. Details regarding the necessary mathematical definitions and concepts are reserved for appendices, presented either at the end of the book or at the end of each chapter. The basic thinking behind this approach is that the key contributions of computational neuroscience are conceptual, and do not rely on a deep understanding of the underlying mathematics. Separating the math allows the ideas to be presented to a wide audience, many of whom do not have extensive computational training. If some readers are interested in the mathematical details, they can find them in the appendices.

In contrast to these approaches, these notes present the neuroscience and mathematical concepts simultaneously. Why? The primary reason is the following. I believe that the most important insights gained from applying computational techniques to understanding the nervous system result from the *process* of translating biological facts into mathematical facts and vice versa. To convey the value of translating ideas back and forth, it is crucial that the math and the neuroscience be presented together. However, an integrated presentation brings with it a host of difficulties. For example, the notes must be presented in two competing voices – the best presentation of the underlying concepts does not always coincide with the best presentation of the mathematics. Therefore the notes sometimes jump from sections that make conceptual sense to those that make mathematical sense and back again. At each of these transitions, there is always a danger of losing the reader. A second major disadvantage is that the amount of mathematics presented must be severely limited. These notes have been written for a one semester course in computational neuroscience. It would be foolish to try to cram 2-3 semesters of college mathematics on top of the conceptual material to be presented.

Overcoming these difficulties requires a degree of discipline on the part of the reader, particularly those readers that have a limited (or rusty) mathematical background. First and foremost, *be patient!* The clearest way to present mathematical concepts is often very abstract. One begins by defining the mathematical objects to be studied, and then goes on to describe the operations to be performed on these objects. These definitions and operations are often confusing at first, since it is only after working some problems and examples that the key concepts become clear. Thus, contrary to the way it appears in text books, mathematics is almost always learned in cycles. It is natural for concepts to be unclear at first; clarity usually comes only after revisiting ideas multiple times. This is why *working problems is crucial* for learning mathematical concepts. In working through a problem set, ideas get applied to a number of examples, requiring them to be revisited a number of times.

A second admonition: if your understanding of the mathematics presented in these notes is less than rock solid, *don't worry!* I'm not expecting you to learn to pass the final exam in a linear algebra course or to know the ins and outs of information theory. I'm only expecting you to learn enough of the mathematics to understand *what* is going on, not necessarily *how* to do it yourself. I cannot emphasize this point strongly enough. When dealing with mathematical concepts there often seems to be an expectation that anything other than a crystal clear understanding is somehow a failure. I encourage students to judge their success in this course relative to the completeness of their knowledge in other survey courses. For example, one hardly expects to remember everything from the mass of facts presented in a typical introductory neuroscience course.

That said, I *am* expecting people to do some mathematics. While it is possible to grasp many of the key concepts in computational neuroscience after only a hand-waving explanation of the underlying mathematics, how these concepts are shaped by their mathematical roots would remain hidden. Since it is important to appreciate the limitations of computational neuroscience as well as its strengths, one has to grapple with some mathematics.

1.2 How to Read These Notes

In an attempt to organize the material and present it from multiple points of view, I have subdivided the material in a number of ways. The main narrative of the notes is organized according to the competing requirements of presenting the mathematics and the neuroscience.

To help keep the reader on track, information that isn't strictly necessary to the main narrative has been separated out into "examples" and "asides." Working through the examples is crucial for adding some flesh to the skeleton of the main ideas. Asides are included to flesh out the presentation even more, although some of them might be considered excess fat by some. Most examples carry the label "biological," "mathematical," or "network." Network examples tend to be in between the other two categories – they're not particularly close to the biology nor do they illustrate purely mathematical idea. These categories are rather loose. Asides are either "biological," "notational," or "historical." I have a particular fondness for the historical asides, since I feel it is important to get a sense of the historical arc of the main ideas.

Since formulating precise definitions is one of the most important aspects of mathematics, important definitions are separated from the main text and given a number. Crucial bits of mathematics are separated into subsections on "Mathematical Formalism." These sections allow the presentation of important mathematical concepts using a more formal mathematical style. Finally, all important terms are presented in **bold** the first time they are used.

At the end of most sections, I have included a number of problems and "key concepts." Key concepts are mostly just pointers to the parts of the section that are most important for understanding the main ideas. These can be used as a quick review, or as starting points for class discussion – if you don't understand one of the key concepts, you should definitely ask for clarification! Problems come in two types. Roughly half have been marked "(E)" for easy. If the problem seems obvious and you think I must be asking for something deeper, you're wrong. These are meant to check that you really did understand the definition in the first place. The other problems should be a bit more challenging and are meant to give you some practice with the main ideas. The problems are a bit uneven and there probably should be more of them. Writing good problems is one of the most difficult aspects of writing a text book.

Much of the above is what I'd like to have happen. These notes will fall short of these ideals, particularly as the semester wears on. We'll see how I can keep up. And speaking of text books, there is a possibility that something based on these notes may be published some day. If you could read them with a pen in hand and make editorial comments that would be much appreciated. I'm particularly interested in hearing about sections that seem confusing or that could be fleshed out a bit more. Any feedback is appreciated.

Finally, these notes assume a working knowledge of basic neuroscience – essentially the material covered in NACS 641 or a similar systems neuroscience course. If you feel that any of the biology needs more explanation, see some of the references below or ask me.

1.3 Supplemental Reading

There are a lot of relevant texts out there. I've looked through many, but by no means all. Here's an informal set of personal recommendations.

A good survey of computational approaches to understanding the brain, is *The computational brain*, by Pat Churchland and Terry Sejnowski (1992). It is a well-written overview of some successes of taking a computational approach to understanding brain function. Chapter 2 can be used as a good bare-bones introduction to some of the basic neuroscience. A good book on neural coding and information theory is *Spikes*, by Fred Rieke, Dave Warland, Rob de Ruyter van Steveninck and Bill Bialek, (MIT Press, 1997). I looked at a number of computational neuroscience text books in preparing these notes. The closest in spirit is *Theoretical Neuroscience*, by Peter Dayan and Larry Abbott, MIT Press, 2001. This covers similar material, but does not really integrate the mathematics into the presentation, and generally assumes a stronger mathematical background. I may use big chunks of this material in the course. At a more biophysical level, I can recommend *The Biophysics of Computation*, by Christoph Koch, Oxford University Press, 1999. This book focuses on information processing in single cells, and much of the presentation is complementary to that presented here. At the network level, I can recommend *Computational Explorations in Cognitive Neuroscience*, by Randy O'Reilly and Yuko Munakata, MIT Press, 2000. I can also recommend *An Introduction to Neural Networks*, by James A. Anderson, MIT Press, 2000. This is "neural networks" textbook written by a psychologist who did some foundational work on associative (Hebbian) learning. It's kind of wordy and is getting a bit dated, but it is quite good.

In terms of basic neuroscience texts, I'm most familiar with Kandel, Schwartz and Jessel, *Principles of Neuroscience* 4 ed., McGraw Hill, 2000, a comprehensive graduate/medical level reference book. At the advanced undergrad level, there is Purves et al., *Neuroscience* 3 ed., Sinauer, 2004. A more introductory text is *Neuroscience: Exploring The Brain*, by Bear, Connors and Paradiso, Lippincott, 2001.

For a basic math reference, I recommend *Introduction to Mathematics for Life Scientists*, by E. Batschelet, Springer 1979. This has a good succinct review of calculus and probability and can also be used as a reference for the linear algebra presented in these notes. A good reference for basic probability and statistics is *Probability and Statistics for Engineering and the Sciences*, by Jay L. Devore, Duxbury press, 2000. A more advanced text on linear algebra and dynamical systems theory is *Differential equations, dynamical systems and linear algebra*, by Morris W. Hirsch (my thesis advisor) and Steven Smale (Academic Press, NY, 1974). A slightly more introductory book on linear algebra is *An Introduction to Linear Algebra* by Strang, Wellesley, 1993. A good practical reference — sort of a cheat sheet of basic results, plus computer algorithms and practical advice on doing computations — is *Numerical Recipes in C*, by W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery (Cambridge University Press).

1.4 What is Computational Neuroscience?

Good question!

- Using computers to simulate and model brain function.
- Applying techniques from computational fields (math, physics) to understand the brain.
- Trying to understand the computations performed by the brain.

1.5 Scope of These Notes

The term computational neuroscience covers a dizzying array of approaches to understanding the nervous system, and to achieve any coherence in constructing a course called "computational neuroscience" requires vast subsections of the field to be excluded. A number of different schemes have been used to divide up the field. Many of these fall under the phrase "levels of analysis." One division according to level of analysis is biological: the brain can be studied at a hierarchy of scales ranging from the cellular and molecular level to the level of small localized circuits in the brain to the level of large-scale brain circuits involving multiple neural subsystems (Fig. **??**).

A second class of scheme is that proposed by David Marr (1982). Marr was making computer models of visual perception, and made the distinction between three levels of analysis: the computational, the algorithmic and the implementational levels of analysis. Roughly, the **computational level** is the most abstract and concerns itself with a description of the problem to be solved, *i.e.* what is the computation that is being performed. Here the object of study are the high-level computational principles involved such as optimality, modularity, etc. The **algorithmic level** of description concerns itself with the structure of the solution, *e.g.* the nature of the subroutines used to perform the calculation. Finally, the **implementational level** concerns how this algorithm is actually implemented in a machine. The difference between the algorithmic and implementational levels is often described as analogous to the difference between a high level programming language like C or Lisp and it's platform specific implementation is machine or assembly language.

Yet another scheme that falls under the "levels of analysis" rubric is the distinction between "top-down" and "bottom-up" approaches to understanding the brain. A **top-down approach** starts at the level of cognitive phenomena and tries to reach "down" to connect these phenomena to specific events taking place in the brain. The **bottom-up approach** starts with biological knowledge about brain cells and circuits and tries to determine how these mechanisms support complex mental phenomena.

Dayan and Abbott (2001) have made yet another tripartite division, dividing computational and theoretical models according to three basic purposes for which they can be used. **Mechanistic models** concern themselves with how nervous systems operate based on known anatomy and physiology. **Descriptive models** summarize large amounts of experimental data, accurately describing and quantifying the behavior of neurons and neural circuits. Finally, **interpretive models** explore the behavioral and cognitive significance of nervous system function, often connecting explaining experimental data in terms of certain theoretical principles.

Each of these schemes for subdividing the field has advantages and disadvantages. In many cases, they can be brought into rough alignment. The topics addressed by these notes, can be best localized in terms of the biology hierarchy - almost all the topics explored fall at the level of neurons or local circuits. The notes are generally "middle-out" although they probably fall closer to the bottom-up rather than the top-down approaches. A truly complete survey of computational neuroscience would probably treat two separate clusters of ideas that are dealt with here in only a cursory manner. The first tradition traces its roots back to by far the most successful model in all of neuroscience: the Hodgkin-Huxley model (**?**). The formalism embodied in their model laid the groundwork for how we understand the electro-chemical events at the heart of how neurons transmit and transform information. So-called compartmental modelling belongs to this tradition. These notes generally reside one level of abstraction up from these models. The other cluster of ideas is somewhat more diffuse, and includes behavioral and cognitive level models of brain function. One tradition that is has particular prominence here is the field known as neural networks or connectionist modelling. These notes will have significant overlap with this field, but in general will be more concerned with biological mechanisms and less concerned with the specific computational and cognitive principles underlying perception, learning and memory.

1.6 The Computational/Theoretical Approach

These notes focus on two basic mathematical approaches taken by computational neuroscientists to broaden our understanding of the relationship between neural activity and behavior. The first centers around concepts of probability theory. These concepts are crucial to making progress in understanding how the brain encodes, performs computations on, and then decodes information gathered from the world outside the cranium. The second approach centers around the idea of a **state space**. This is a somewhat abstract notion in which a list of the many variables describing a given neural system are viewed as a single point in some (usually large dimensional) "space." For example, one could assume that a list of the activity level of all the nerve cells within a given region of the brain gives a reasonable characterization of the "state" of that brain region. The set of all possible combinations of firing rates represents the "space" in which such a state lives, and any particular state is represented as a point in that space. The most important contribution of this idea is that the internal variables describing many individual components of a system are combined into a single conceptual "object." It is then an easy conceptual leap to think of one state affecting another, or to view changes in the system over time as mapping out a path in state space, all without getting enmeshed in the particular changes in each of the parts. Computationally, tools exist for analyzing many such systems, particularly if they are **linear**. Thus, the opening chapters of these notes will focus on basic notions in linear algebra, although we'll address certain nonlinearities as well.

Chapter 2

Neural Coding

Much of the first part of these notes concerns itself with the topic of **neural coding**. The phrase "understanding the neural code" has great intuitive appeal, but nailing down exactly what such an understanding would entail can be a slippery proposition. Taken at its broadest interpretation, saying that one is interested "cracking the neural code" is no more specific than saying that one is interested in understanding how the brain works. While more exact definitions could be attempted, I will associate "neural coding" with attempts to analyze brain function focusing on the question of neural representation, *i.e.* how the brain represents and transforms information about objects in the environment.

2.1 Nobel Prize Winning Research - Lord Adrian

The most natural starting point for a discussion of neural coding is the ground breaking experiments carried out in the 1920's by **E.D. (Lord) Adrian** (1891-1977). Adrian was the first to employ instruments sensitive enough to record from single axons of sensory receptor neurons (previous recordings were from nerve bundles). The first neurons that he recorded from were stretch receptors in the muscle of the frog, but he soon recorded from a range of receptor neurons responding to touch, as well as visual neurons in the eel (Adrian, 1964).

Adrian's results are so ingrained in the culture of neuroscience, that it is difficult to appreciate just how important these Nobel prize-winning results (1932) were. First and foremost, Adrian demonstrated that information about the world enters the nervous system as a series of pulses, whose size and shape depended only on the local conditions in the axon (see figure 2.1 for an example of a modern recording). Therefore, information is carried only by the temporal pattern of impulses, rather than the type or shape of the impulses themselves. Therefore, information is carried only by the temporal pattern of impulses, rather than the type or shape of the impulses themselves. These impulses are commonly referred to as **action potentials** or **spikes**. These are self-regenerating electrical events that travel down axons and eventually trigger release of chemical neurotransmitters at outgoing synapses. One of the key properties of spike propagation is that information can be reliably transmitted over long distances. While subsequent experiments have revealed that some neurons communicate without generating action potentials, spiking is the dominant form of communication in the brain.

Second in importance to the discovery that the shape of the action potential did not appear to carry significant information was Adrian's finding that as the magnitude or intensity of stimulation was increased, the sensory neurons produced action potentials at an increasing rate (figure 2.2). Thus, at least to a first approximation, the nervous system seemed to use firing rate to encode information about the world. This rate encoding hypothesis has dominated neuroscience ever since. However, it is quite possible that information is encoded in the *pattern* of spike timing rather than (or in addition to) their overall rate of production. When diving into the debate over "rate

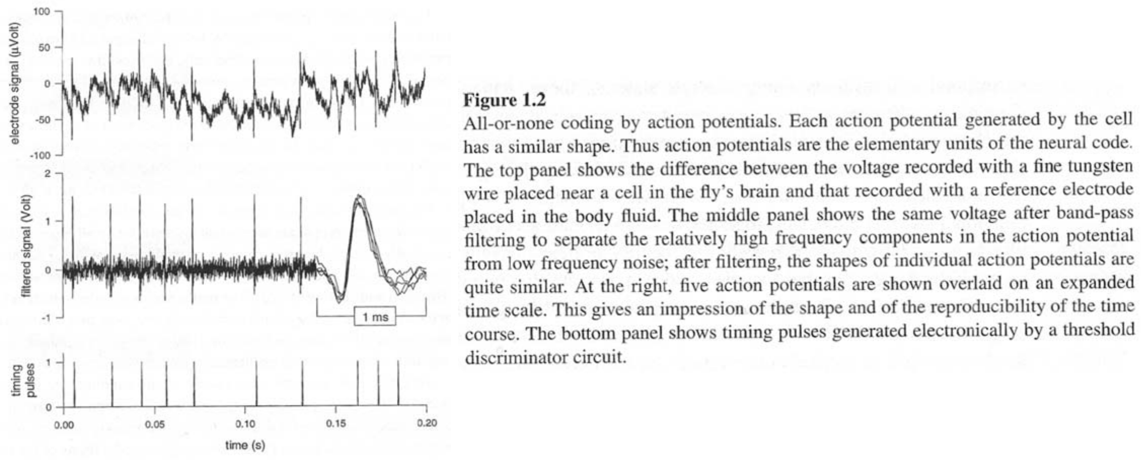


Figure 1.2
All-or-none coding by action potentials. Each action potential generated by the cell has a similar shape. Thus action potentials are the elementary units of the neural code. The top panel shows the difference between the voltage recorded with a fine tungsten wire placed near a cell in the fly's brain and that recorded with a reference electrode placed in the body fluid. The middle panel shows the same voltage after band-pass filtering to separate the relatively high frequency components in the action potential from low frequency noise; after filtering, the shapes of individual action potentials are quite similar. At the right, five action potentials are shown overlaid on an expanded time scale. This gives an impression of the shape and of the reproducibility of the time course. The bottom panel shows timing pulses generated electronically by a threshold discriminator circuit.

Figure 2.1: Taken from Rieke et al. (1997).

codes" and "temporal codes" things get pretty murky rather quickly. We will discuss some of these issues in the section on temporal coding, but to start we will explore the relationship between spike trains and spike rates in some detail.

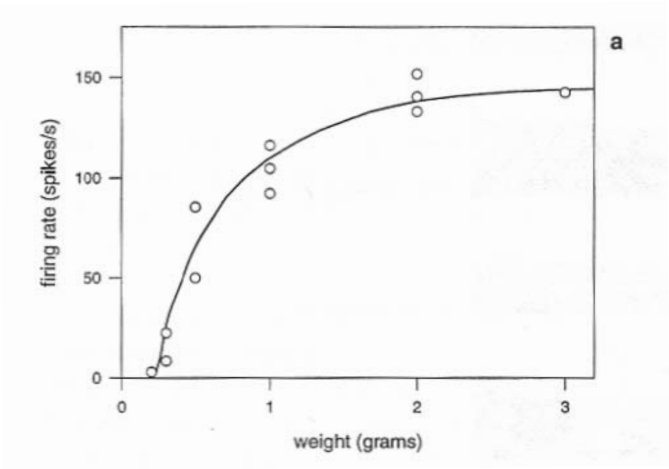


Figure 2.2: Average firing rate of a stretch receptor as a function of the weight applied to the muscle. Experiment performed by E.D. Adrian; figure taken from Rieke et al. (1997).

2.2 From Spikes to Rates

We will start by exploring different ways of assigning spike rates to a given take a series of action potentials, or **spike train**. In particular, suppose we have a list of action potential occurrence times, $\{t_1, t_2, \ldots, t_i, \ldots\}$ (we often simply write $\{t_i\}$ and say that t_i is the time of the ith spike). How do we come up with a meaningful definition of the rate at which these spikes are produced? Certainly any notion of spike rate should have the units of spikes per unit time, *i.e.* it should somehow be calculated as a fraction:

$$spike\ rate = \frac{\#\ of\ spikes}{period\ of\ time} \tag{2.1}$$

2.2.1 Windowing

The most obvious way to define rate is to decide on fixed a period of time, or **time window**, and count the number of spikes. For example, if one is interested in how a visual neuron responds to bars of different orientations, one can present such stimuli for a fixed period of time, and count spikes during the period in which the stimulus was shown. In other experiments, one is interested in how the spike rate changes over time. A simple way to calculate the rate in this case is to divide the experiment into short segments or "bins," and count the number of spikes in each **time bin** or window (figure 2.3A). (The representation at the bottom of each plot in which time of each spike is represented with a mark - in this case a vertical line - is known as a **spike raster** plot.) One difficulty with this approach is that one can get different answers depending on where the spikes fall relative to the edges of the time bins. These "edge effects" can be eliminated by using a **sliding window**, *i.e.* letting the window slide along the time axis, assigning the rate at any time to be the number of spikes in a window centered at that time divided by the length of the window (figure 2.3B). The location of the edges then depend on the spike times themselves. Note that this procedure is exactly equivalent to the procedure of placing a "window" centered on each spike whose height is equal to 1 divided by the window width, and determining the rate by summing up the heights of all the windows overlapping that point in time (problem 2.4.2). Using a sliding window allows one to consider windows of arbitrary shapes. Often, one takes a smooth window, eliminating the sudden jumps in firing rate (figure 2.3C).

In choosing a window to define the spike rate, one faces a fundamental tradeoff in estimating the underlying rate: variability can be reduced by using large windows and averaging over many spikes, but this averaging smoothes over rapid fluctuations in spike rate. This tradeoff is demonstrated in figure 2.4. Spikes were generated at random from a rate function that makes a jumps from $50 spikes/sec$ to $200 spikes/sec Hz$ at $500 msec$. A large window (figure 2.4A) leads to a good estimate of the firing rate, but smooths out the transition from low to high rate. A short window (figure 2.4B) can resolve the transition, but gives a noisy estimate of the firing rate.

2.2.2 Spike Trains as Rate Functions

Thus far, we have represented spike trains and spike rates in fundamentally different ways. Spike trains are a list of spike times $\{t_i\}$, whereas spike rate is a function of time, rate $= \mathbf{r}(t)$ (figures 2.3 and 2.4). It will be convenient to be able to represent both types of objects using a single mathematical framework. Our approach will be to write the spike train as a function of time, *i.e.* to come up with a rate function $s(t)$ such that spikes are viewed as a very brief increase in the firing rate, and the period between spikes has zero spike rate. How high should the rate function go at the time of each spike? Intuitively, we can make the increase in rate around a spike to be infinitely short, as long as the rate function becomes infinitely large over that period. This process is shown in figure 2.5. Note that when constructing a windowed rate in this manner, the "hump" placed around each spike should have a total area equal to 1. This is because one goes from spike rate to spike number by integrating over time, *i.e.* the number of spikes in the time interval from a to b should be equal to the integral $\int_a^b dt\ \mathbf{r}(t)$.

For an idealized spike train, we would to place an infinitely thin, infinitely tall window to mark each spike. Such a hump placed at the time $t = 0$ is known mathematically as a **Dirac delta function**, $\delta(t)$. It follows that $\delta(t - \hat{t})$ represents a delta function centered at $t = \hat{t}$. Note that the delta function is not a true function, since it doesn't take on a specific value when $t = 0$. However, such a generalized function can be defined rigorously as the limit of the process depicted in figure 2.5, and it can be treated just like a function that has the following properties:

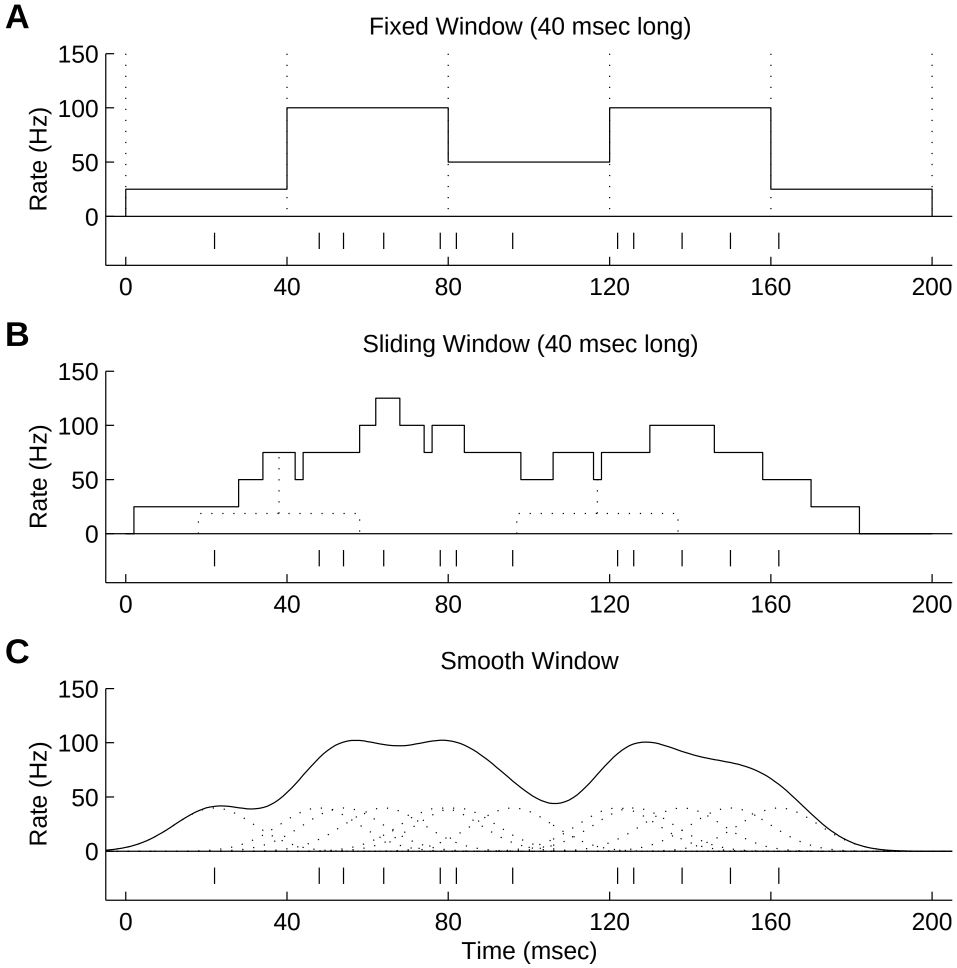


Figure 2.3: Different windowing strategies for defining spike rate.

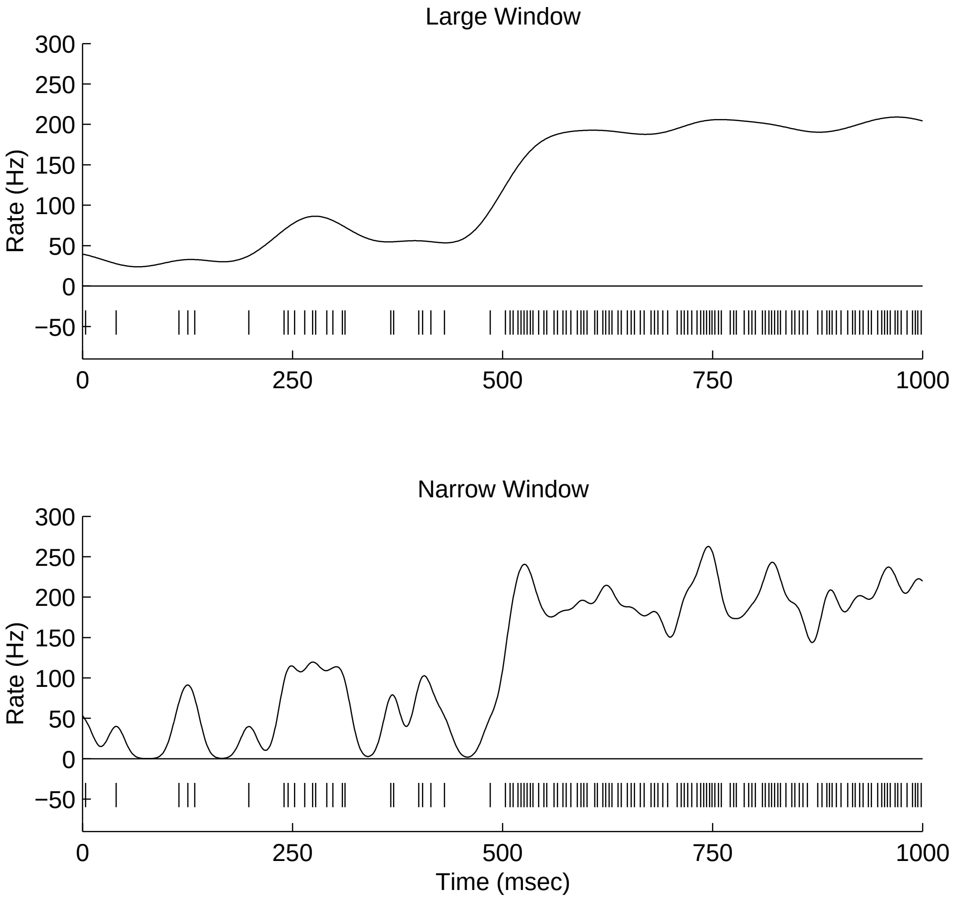
1. $\delta(t) = 0$ for $t \neq 0$.
2. $\int_a^b dt\ \delta(t) = 1$, for $a \leq 0 \leq b$.
3. $\int_a^b dt\ f(t)\delta(t) = f(0)$, for any function $f(t)$ and for $a \leq 0 \leq b$.

With this definition we can represent the spike train $\{t_i\}$ as the sum $s(t) = \sum_i \delta(t - t_i)$.

No that we can view spike trains as a rate function, the windowed rate function can be seen as a "smoothing" of this rate function using the function describing the window. Mathematically, this smoothing process is called a convolution. The **convolution** $f * g$ of two functions f and g is defined as follows:

$$f * g(t) = \int_{-\infty}^{\infty} ds f(s) g(t - s) \tag{2.2}$$

To see how the convolution works as a smoothing operation, let $g(t)$ be the function to be smoothed and let $w(t)$ be a rectangular window function that is $25msec$ wide. To operate as a smoothing window, w must have a total area equal to 1, so it must be $1/(25msec) = 1000/25sec^{-1} = 40Hz$ tall, *i.e.* $w(t) = 40Hz$ for $12.5msec \leq t \leq 12.5msec$, $w(t) = 0$ otherwise. Now we find $w * g(50)$,


Large Window

Narrow Window

Figure 2.4: The tradeoff between large and narrow windows.

the value of the smoothed version of $g(t)$ evaluated at $t = 50msec$:

$$\begin{aligned} w * g(50msec) &= \int_{-\infty}^{\infty} dsw(s)g(50msec - s) && (2.3) \\ &= \int_{-12.5msec}^{12.5msec} ds(\frac{1}{25msec})g(50 - s) && (2.4) \\ &= (\frac{1}{25msec}) \int_{-12.5msec}^{12.5msec} dsg(50 - s) && (2.5) \\ & && (2.6) \end{aligned}$$

The integral term represents the area under the function g from $50 - 12.5 = 37.5msec$ to $50 + 12.5 = 62.5msec$. The term out front multiplies this area by 1 over the length of the interval. Therefore, the value of the convolution $w * g(50msec)$ is equal to the average value of g in the $25msec$ surrounding $t = 50$. An example of this smoothing is shown in figure **??**A.

Now suppose that $g(t)$ is a spike train, *i.e.* $g(t) = \sum_i \delta(t - t_i)$. Then $\int_{-12.5msec}^{12.5msec} ds(\frac{1}{25msec})g(t-s) = \sum_i \int_{-12.5msec}^{12.5msec} ds(\frac{1}{25msec})\delta(t - t_i - s)$. Since integrating the product $f(s)\delta(t - t_i - s)$ just picks out the value $w(t - t_i)$ (see item 3 above), the convolution $w * g(t)$ gives $n/(25msec)$ where n is the number of spike times t_i that fall within a $25msec$ interval around t. That is, $w * g$ represents the windowed firing rate for the window function w (see figure **??**A).

The same intuition holds if w is no longer a rectangular window. Instead of the exact average

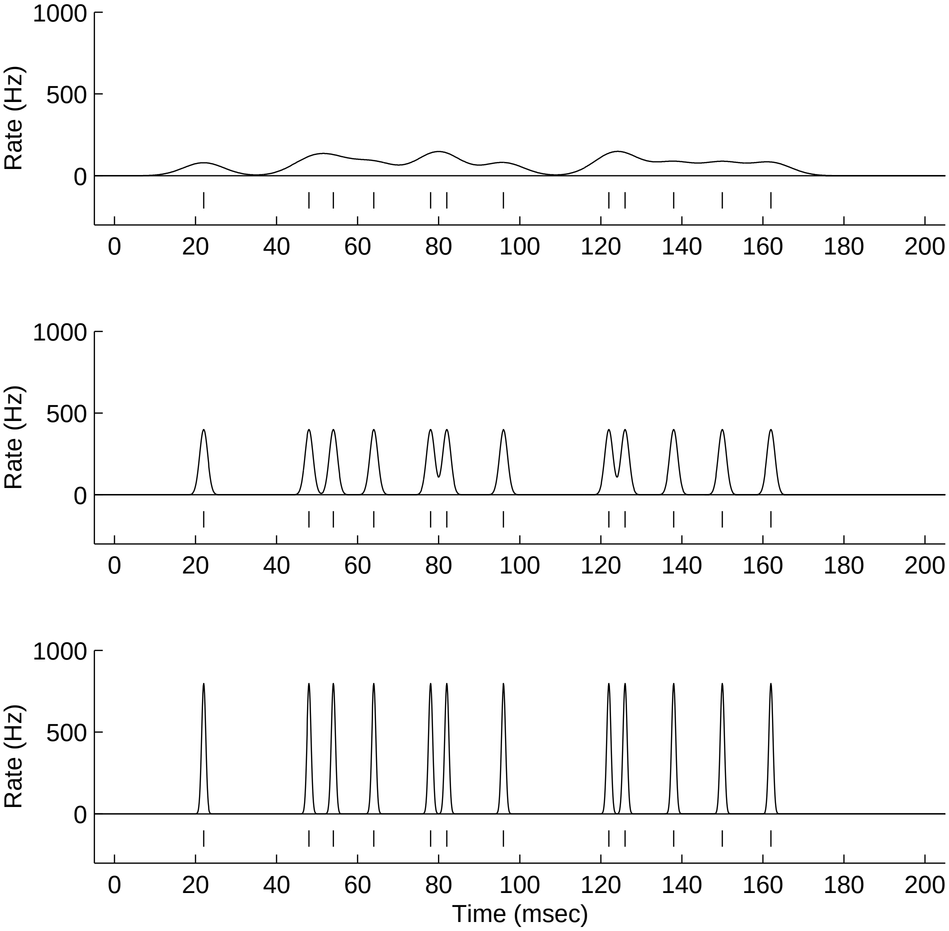


Figure 2.5: Representing a spike train as a rate function with increasingly narrow windows.

over an interval, the convolution represents a weighted average of $g(t)$ where the relative weighting is determined by the shape of w (see figure **??**B,D).

2.2.3 The ISI Rate

An alternative to using a fixed time window is to fix the number of spikes and then measure the time. In particular, one can focus on the time between two-adjacent spikes, and define the spike rate during that period of time, as 1 divided by the length of the interval (figure 2.7). Since the time between spikes is referred to as the **interspike interval (ISI)**, I will refer to spike rates calculated in this way as the **ISI rate**. When using the windowed rate, time scale at which rate changes can be measured are determined by the window. In contrast, when using the ISI rate, the rate is defined on a time scale that is determined by the spike train itself.

2.2.4 The PSTH Rate

As shown in figures 2.4B and 2.7, if only a few spikes contribute to the determination of the spike rate at any given time, then the estimate of the rate can be quite variable. One way to make sure that many spikes contribute to the rate calculation is to use a large window, as in figure 2.4A. However, large windows have the problem of being unable to capture rapid changes in firing rate.

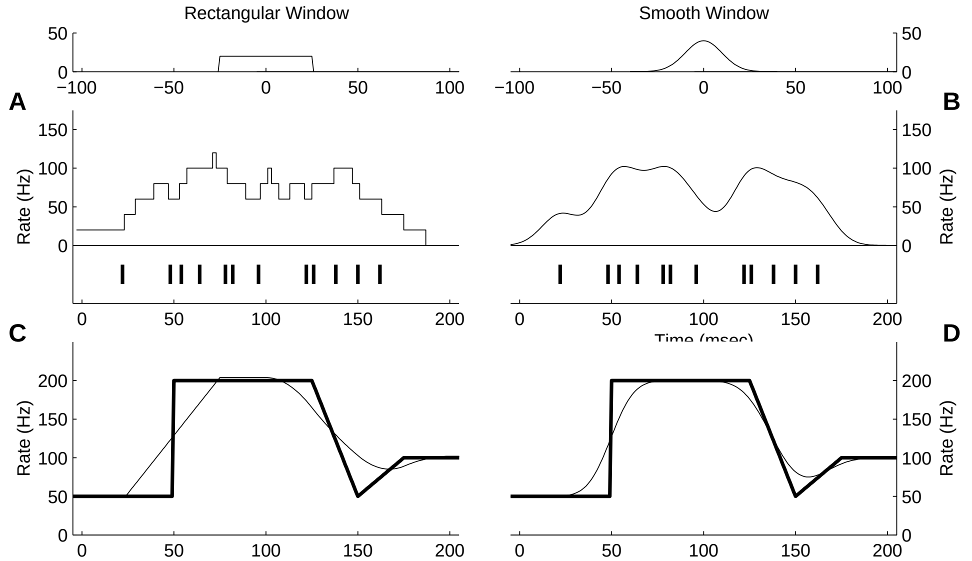


Figure 2.6: Convolution of the windowing function shown in the inset with a spike train (A,B) and a continuous function (C,D).

To reliably capture these rapid transitions, one needs more spikes. One way to get more spikes is to perform repeated trials of the same experiment. For example, figure 2.8 shows the result of repeated trials of ... Just like in the windowed rate, the time axis is divided into bins and a histogram is made of the number of spikes in each bin. Such a histogram is known as a **peri-stimulus time histogram** or **PSTH**. By dividing by the width of each bin, the PSTH can be expressed as a rate. I will refer to the rate calculated in this way as the **PSTH rate**. The PSTH rate is essentially the same as a windowed rate and has issues with edge effects and tradeoffs between variability and temporal precision. However, if a sufficient number of spikes are collected, these problems become minor and one can accurately measurement spike rate at a time scale that is significantly smaller than the typical interspike interval.

The PSTH rate is sometimes criticized as being merely a convenient data analysis tool for estimating the underlying spike rate, but telling us nothing about the neural code. The argument goes that an animal must respond to a single presentation of the stimulus; it cannot wait for repeated presentations to create a histogram. The obvious response to this criticism is that the brain contains lots of neurons. Therefore, if we assume that there is a whole population of neurons whose response properties are the same as the neuron being recorded from, then the animal could average across neurons to get something like a PSTH in a single trial. While this response addresses the original criticism of the PSTH, it seems to require that neurons must be organized into groups with similar response properties. In chapter 5 below, we will show that the only restriction placed on the organization of neural circuits by this point of view is that a given stimulus must activate many neurons.

2.3 From Rates to Spikes

So far we've discussed how to assign a rate function to a given spike train. When performing theoretical computations related to neural coding or when making models of neural systems it is not uncommon to find oneself in the position of having to specify how spikes are produced by a

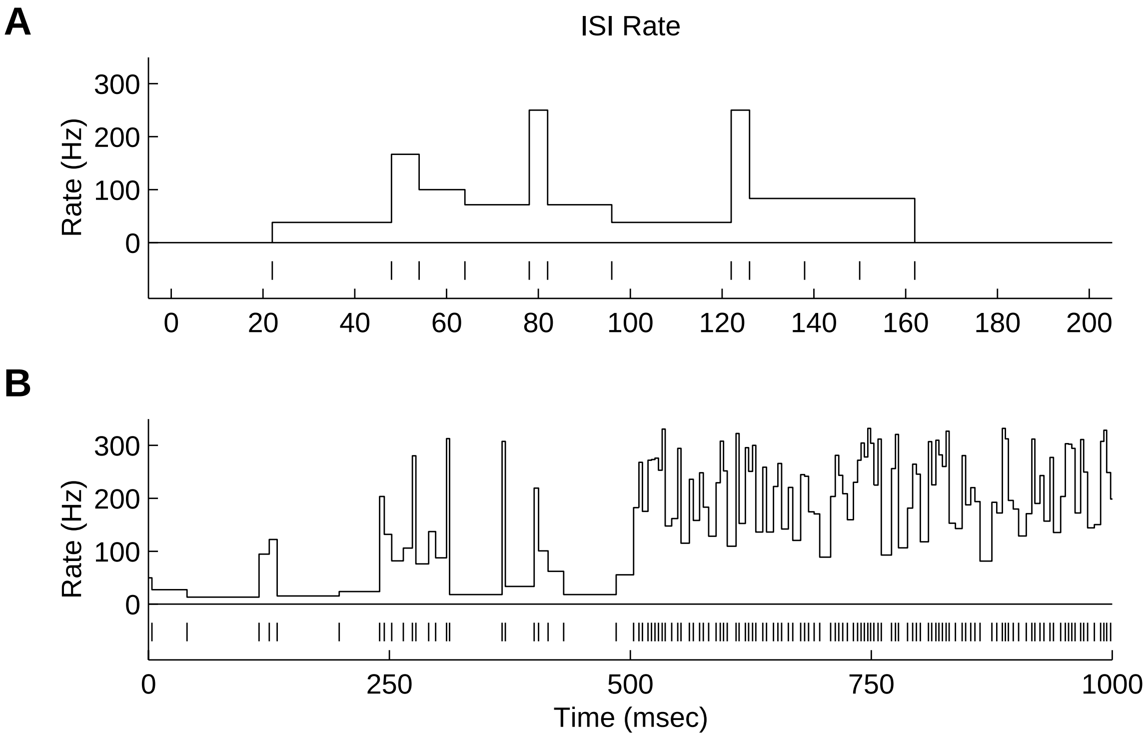


Figure 2.7: Defining rates 1 divided by the interspike interval (ISI).

neuron firing at a given spike rate. We will explore some of the biological issues related to spike generation in chapter **??**, but for now we introduce two simple phenomenological models of spike generation. It is useful to think of these two models as coming from “inverting” the definitions of the ISI and PSTH rates. We then introduce a more realistic model that we will explore in more detail in chapter 5.

2.3.1 The Poisson Neuron

Consider a neuron firing at a constant rate, say $r = 20\ spikes/sec$. Then one would expect that the PSTH collected from many trials, would be constant, *i.e.* in any small time period Δt, the neuron will produce $r\Delta t$ spikes when averaged over many trials. From this perspective, we can think of the rate at any given time as proportional to the probability of producing a spike at that time. Moreover, in combining spikes from many different trials, the PSTH “averages out” information regarding the dependency of one spike on the next in a given trial. By assuming no dependency, one can construct a simple and useful formal model of spike production, the **Poisson neuron**. A Poisson neuron produces a random sequence of action potentials that have two important properties:

- The probability of producing a spike at any given time is give by rate function $\mathbf{r}(t)$.
- Spikes are produced independently, *i.e.* the probability of producing a spike is not affected by the presence or absence of other spikes.

The last property is an mathematical idealization of the biology. For example, spikes have a finite width and so after the start of one spike means that there can't be another spike for at least this duration. Moreover, after each spike real neurons have a **refractory period** where the probability of generating a spike is reduced. MORE.

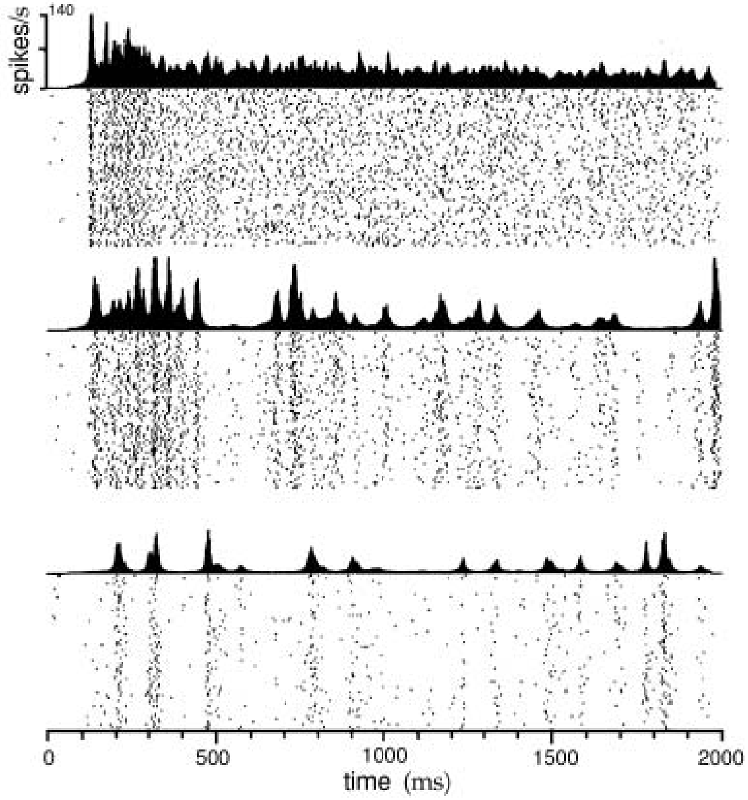


Figure 2.8: The PSTH rate. Responses of a single neuron in visual area MT of a macaque monkey to repeated trials of three different moving random dot stimuli. Dots represent individual spike times, and spike histograms are plotted below each raster plot. (Adapted from Bair and Koch, 1996; taken from Dayan and Abbott (2001).)

Mathematical Aside. This model derives its name from the mathematical concept of a **Poisson process**. A **stochastic point process** is a mathematical process that generates a sequence of events with some probabilistic structure. A Poisson process is a stochastic point process that has the properties 1 and 2 above. One can show that the probability of getting n spikes in any given interval equal to $e^{-\lambda}\lambda^n/n!$.

2.3.2 The Perfect Integrator

Consider a neuron firing at a constant rate, say 20 $spikes/sec$. From the perspective of the ISI rate, a constant rate would mean a constant interspike interval ($1/20sec = 50$ $msec$ in this case). A simple model of a neuron that has this property is the **perfect integrator**: the model neuron adds up (integrates) its inputs and then produces a spike whenever this integrated input reaches some threshold value, at which time the process starts over (figure 2.9). For obvious reasons, this type of model is referred to as an integrate-and-fire (IF) neuron. The rate at which the input arrives is proportional to the slope of the integrated input – the greater the input, the faster that the model returns to threshold and the greater the firing rate.

EXPLAIN FIG.

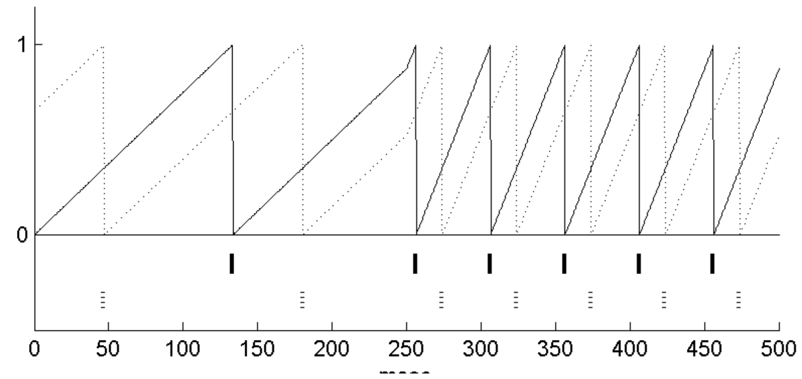


Figure 2.9: Spike trains generated by a perfect integrator. Spike rate is 15 Hz from 0 to 250 $msec$, and 40 Hz thereafter. Solid and dashed lines represent different initial conditions.

2.4 A Simple Model Neuron

The above models are phenomenological models for how a neuron might produce spiking output at a given rate. They don't pretend to model how actual neurons convert synaptic input into trains of action potentials.

EXPLAIN FIG.

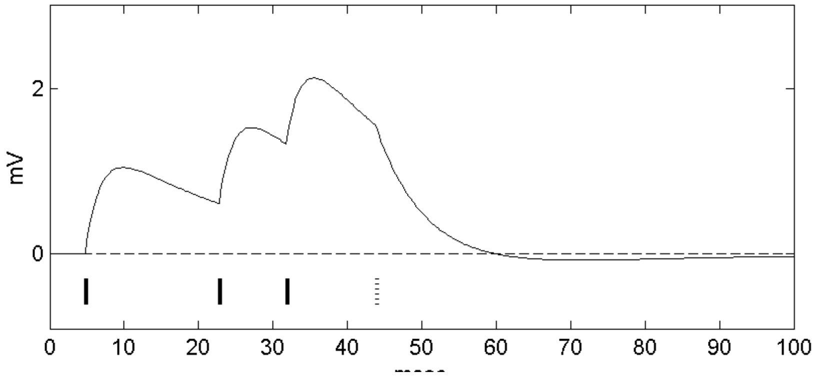


Figure 2.10: Summation of **postsynaptic potentials (PSPs)**. The figure shows three short excitatory PSPs (EPSPs; times marked by solid lines), followed by a long and large inhibitory PSP (IPSP; dashed line).

Problems

Problem 2.4.1 Show with a picture that calculating the rate using a continuously sliding rectangular window is equivalent to placing a window centered on each spike and adding.

Problem 2.4.2 Use the definition of a convolution to show that the windowed spike rate for the spike train $\{t_i\}$ can be written $\mathbf{r}(t) = \sum_i f(t - t_i)$, where $f(t)$ is the windowing function. This says that the process of convolving a window function and spike train is equivalent to centering a window function at each spike time and adding.

Problem 2.4.3 Suppose that $g(x)$ is a linear function, $g = mx + b$, and $f(x)$ is symmetric window function, $f(x) = f(-x)$ and $\int_{-\infty}^{\infty} dx f(x) = 1$. Show that $f * g(x) = g(x)$. *[Hints: 1. Try to figure out why this would be true if f is a square window. 2. Break the integral $f * g(x) = \int_{-\infty}^{\infty} ds f(s) g(x-s)$ into $\int_{-\infty}^{0} ds f(s) g(x-s) + \int_{0}^{\infty} ds f(s) g(x-s)$.]*

Problem 2.4.4 Suppose that $g(x)$ is a linear function, $g = mx + b$, and $f(x)$ is symmetric window function, $f(x) = f(-x)$ and $\int_{-\infty}^{\infty} dx f(x) = 1$. Show that $f * g(x) = g(x)$. *[Hints: 1. Try to figure out why this would be true if f is a square window. 2. Break the integral $f * g(x) = \int_{-\infty}^{\infty} ds f(s) g(x-s)$ into $\int_{-\infty}^{0} ds f(s) g(x-s) + \int_{0}^{\infty} ds f(s) g(x-s)$.]*

2.5 Receptive Fields, Response Functions, and Tuning Curves

At the most basic level, all experiments designed to uncover the neural code can be described as presenting a bunch of stimuli (more or less under the control of the experimenter) and recording the neural responses. Although we will describe some systematic methods below, most often the stimulus set is arrived by a combination of educated guesses and trial and error.

The most common form of presenting the results is in the form of a **response function** – a graph where the mean number of spikes is represented as a function of the particular parameter value associated with each stimulus. For example, the left part of figure 2.11 shows the contrast response function for a cell in the primary visual cortex, and the right plot shows an orientation tuning curve. MORE.

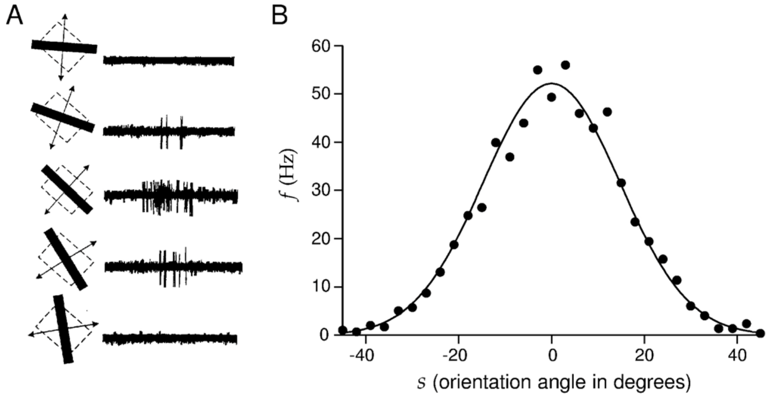


Figure 2.11: Orientation tuning of neurons in the visual cortex. A shows extracellular recordings in a monkey stimulated by moving bars. B shows the average firing rate of a neuron in the primary visual cortex of the cat as a function of the angle of a light bar stimulus. Taken from Dayan and Abbott (2001).

2.6 The Dominant Paradigm

There is a particular view of brain function that is implicit in many discussions of neural coding. In fact, this viewpoint dominates much of neuroscience. In this view, organisms are optimized to extract information presented to them by their environment and, based on this information, select behaviors that optimize their chance of survival. This point of view is depicted in figure 2.14. Information first enters the brain through sensory receptors. Information is extracted via a series of (sensory) processing stages. Based on this information, and perhaps information recalled

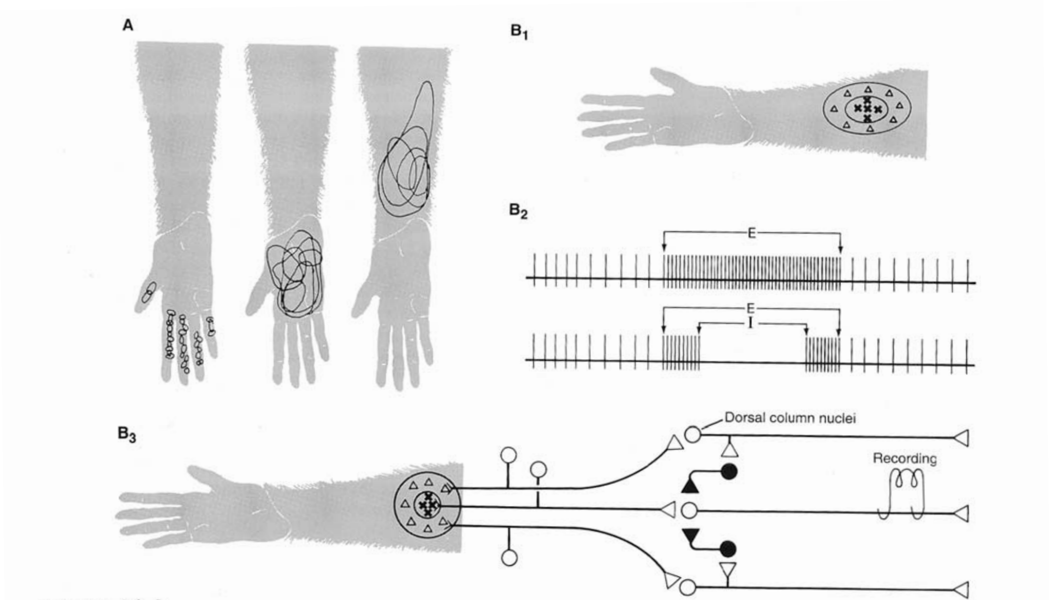


FIGURE 26–8
Fine structure of the receptive field of mechanoreceptors.
A. The size of the receptive fields of mechanoreceptors varies throughout the body. The fields are small in the distal finger tips; they become larger in the hand and even larger in the forearm.
B. 1. Idealized relationship of the excitatory (x) and inhibitory (Δ) zones of the receptive field of a neuron in the postcentral gyrus. In this example a larger zone of inhibition surrounds the excitatory zone. **2.** Extracellular recordings from a single cell in the cortex illustrate the dynamics of the excitatory (**E**) and inhibitory (**I**) zones of the receptive field. A stimulus applied steadily to the excitatory part of the receptive field elicits a steady firing in the cell (**trace 1**). When the inhibitory region on the skin is stimulated along with the excitatory part, the excitation is inhibited. When the inhibitory stimulus is removed, the excitatory stimulus is again effective (**trace 2**). **3.** Simplified model of the input connections for a cell in the dorsal column nucleus (electrode) with a receptive field that has an excitatory region and inhibitory surround. A stimulus applied on the skin to the center of the receptive field activates receptors that excite one group of cells in the nucleus (the excited region or discharge zone). Stimulation of surrounding skin activates other receptors that end on inhibitory interneurons and suppress the firing of the cells activated by the excitatory center of the receptive field.

Figure 2.12: Somatosensory receptive fields. Taken from **?**.

from memory, the organism makes a decision to act. Finally, the details of an appropriate motor command are computed in another series of (motor) processing stages, eventually leading to behavior.

This picture is the starting point for a number of the most basic controversies about how we view the actions of the nervous system. For example, by depicting various aspects of brain function as separate (sensory processing, decision making, memory, and motor processing), this picture implies a modular view of the nervous system. Controversies related to the localization of brain function have raged since the days of the phrenologists in the nineteenth century, and continue in fights over the interpretation and importance of the latest brain imaging studies. Figure 2.14 has also been criticized as containing a **humunculus** or “little man” inside the box labelled “decisions, consciousness, etc.” Segregation of these executive functions from the more mundane jobs of sensory, motor and memory processing, avoids the “hard” (and important) questions relating to the relationship between brain and mind (perception, consciousness, emotions, etc.). Of course this separation is often a practical necessity in order to be able to approach a host of more mundane but devilishly complex questions about the way the brain works.

Another criticism of that can be levelled against the dominant view, is the fact that information flows only in one direction, originating with the stimulus and ending with the response. Since it allows for complex interaction with internal representations and memory, the dominant viewpoint escapes some of the criticisms levelled at the strict stimulus-response paradigm adopted by the behaviorist school in the early to mid twentieth century. Nevertheless, the action in figure 2.14

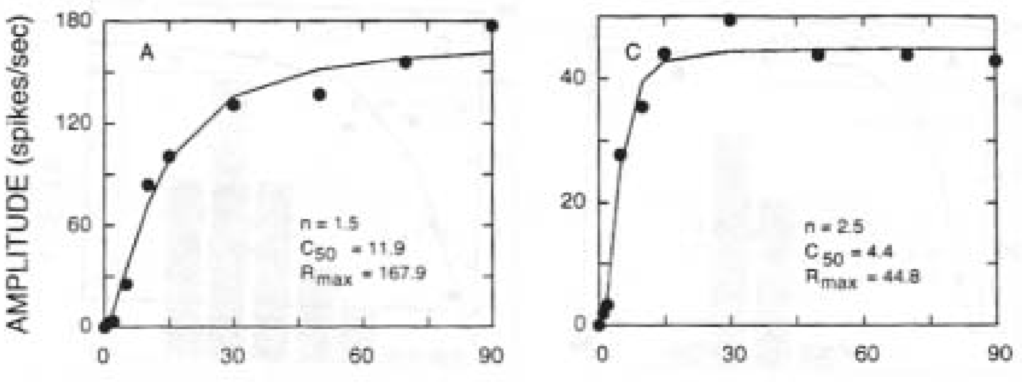


Figure 2.13: Response functions for two representative neurons recorded within the cat visual cortex when stimulated with moving grating patterns of different contrast. The parameters are those for the best fit for the equation of the form $R(C) = R_{max}C^n/(C^n + C_{50}^n)$. Taken from Albrecht (1995).

starts with the stimulus and ends with a response. This assumption has often been criticized as a much too passive view of an organisms role in the world. Some have argued that action should be emphasized to a much greater extent, with the behavior of the animal determining to a large degree which stimuli are experienced. This viewpoint is reflected in current trends toward "active perception." Taking a more extreme view, one could reasonably argue that the motor end of the picture is fundamental – animals have evolved to act. Sensory input is of course important, but instead of looking to the world to determine the origin of behavior, perhaps on should view sensory stimuli as making minor adjustments to an animal's ongoing behavioral repertoire. These issues touch on two of the basic dichotomies encountered while studying the brain: to what degree is behavior innately specified vs. learned, and to what degree is a given pattern of brain activity driven by the external stimulus vs. internal brain processes.

2.7 A Survey of Neural Codes

There are many ongoing and often heated debates related to issues of neural coding. Often these are presented as either/or debates between mutually exclusive notions of the neural code. Does the brain use a firing **rate code** or a **temporal code**? Does it use a **population code** or a **local code**? Are population codes generally based on **coarse coding** or **sparse coding** principles? (These terms will be defined when we explore the relevant issues in more detail.) As one digs a bit deeper, these clear dichotomies often get rather murky, and their relevance for understanding the brain can get lost. One important source of confusion is that the relevance of the different coding paradigms often depends to a great deal on the particular experiment.

2.8 Limitations of the Rate Coding Hypothesis

2.8.1 Labelled Line Encoding

These results raised a number of questions. Most fundamentally they raised the question of why we experience the world as a continuous flowing scene of continuous objects, when the information that enters the brain is a complicated pattern of discrete impulses. Moreover, the impulses caused by visual stimuli look identical to those caused by somatosensory stimuli. How come one set of

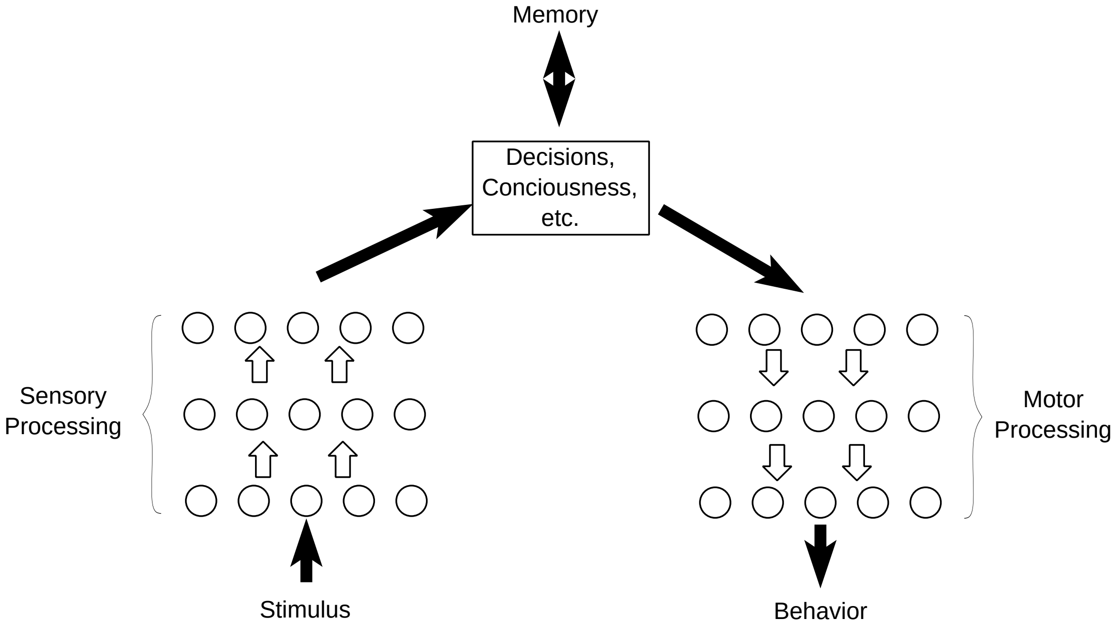


Figure 2.14: The dominant paradigm for thinking about brain function.

impulses leads to the experience of seeing and the other to the experience of touch? The obvious answer to the second question is that the visual experiences are caused by impulses arising from the eye and touch sensations are caused by impulses originating in the skin. To quote Adrian, "the quality of the sensation seems to depend on the path which the impulses must travel, for apart from this there is little to distinguish the message from different receptors" (Adrian, 1964). ignore We will discuss this "labelled line" code in more detail on the neural coding section of these notes. I should warn you that some recent experiments call into question the labelled line explanation for the assignment of neural impulses to their appropriate sense experience.

While the explanation of labelled lines may seem obvious and direct, it relies on a fairly static notion of "path" or "line." However, over the past 20 years or so a number of experiments have demonstrated that the brain is capable of amazing feats of reorganization, even in the adult. Of particular importance are a series of experiments performed by Mriganka Sur and colleagues (). By ablating the auditory areas of the thalamus and the primary visual cortex at the appropriate point in developments, they were able to get neurons in the visual thalamus to send their axons to what would normally be the auditory cortex. Physiological and behavioral experiments revealed that these "auditory" cortical neurons responded selectively to oriented visual stimuli, organized themselves into visual maps, and could be used to guide visually directed behavior.

The rewiring experiments do not really contradict the labelled line idea, since it is the entire path that constitutes a labelled line. Presumably, neural pathways "downstream" of the auditory cortex were also rewired so that the previously auditory neurons connected with the "decision" and "motor" circuits guiding visual behavior. However, these experiments do point to the possibility of a circularity in Adrian's labelled line argument. For if no fixed pathway can be associated with visual perception, then one cannot use the pathway argument to delineate "visual" from "auditory" spikes. Turning the argument on its head, one could argue that the visual pathway should be defined as the set of neurons whose activation contributes to visual perception. But this begs the question of how neural activity leads to perception, and leads straight to the formidable philosophical question of the relation of mind and brain. Adrian was quite frank in acknowledging that his experiments didn't really have much to say about this age-old problem.

2.8.2 Adaptation

Adrian's third basic finding was that neural responses **adapt**, *i.e.* the initial presentation of a stimulus causes a neuron to produce spikes at a certain rate, but as the neuron "adapts" to this stimulus its firing rate slows (fig. 2.15). The most basic lesson to draw from this result is that the representational power of sensory neurons may be focused on representing *changes* in the state of the environment, rather than the state of the environment *per se*. As pointed out by Adrian, adaptation causes a significant complication for the notion of a rate code, because it removes the one-to-one relationship between spike rate and magnitude of the stimulus. For example, a stretch receptor that is firing at XX spikes per second[1] could be the immediate result of lifting a YY pound weight, or could result from a YY pound weight that was hoisted a second ago. Thus, despite Adrian's results stretch receptors cannot be said to use firing rate to *represent* the weight of an object, at least not using the most direct notion of rate encoding. It is surprising how often these basic facts have been ignored when designing and/or interpreting experiments aimed at discovering the neural code.

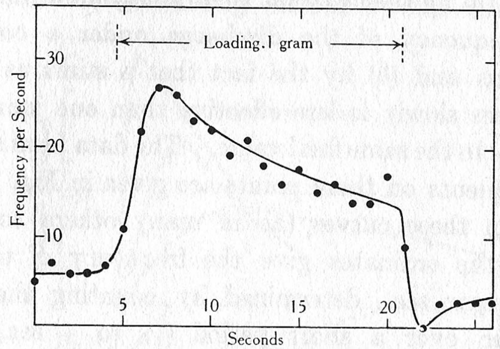


FIG. 12. FROG'S STERNO-CUTANEOUS PREPARATION. CURVES SHOWING THE FREQUENCY OF IMPULSES PER SEC. PRODUCED BY STRETCHING THE MUSCLE.

Figure 2.15: Adaptation in a frog stretch receptor. Taken from (Adrian, 1964).

2.9 Temporal Coding

2.9.1 Pulse Codes

2.9.2 Synfire Chains

We may get to synfire chains later in the course. Otherwise this might serve as a decent project topic.

---

[1] Events per second have units of Hertz ($Hz = sec^{-1}$, *i.e.* 1 Hz denotes one event per second.

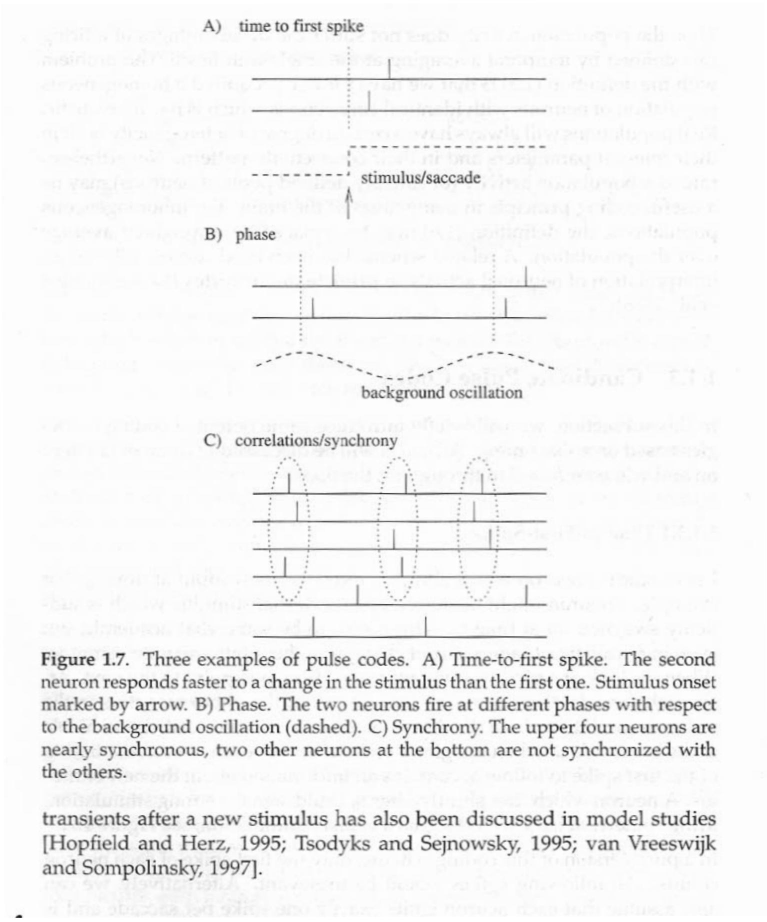


Figure 1.7. Three examples of pulse codes. A) Time-to-first spike. The second neuron responds faster to a change in the stimulus than the first one. Stimulus onset marked by arrow. B) Phase. The two neurons fire at different phases with respect to the background oscillation (dashed). C) Synchrony. The upper four neurons are nearly synchronous, two other neurons at the bottom are not synchronized with the others.

transients after a new stimulus has also been discussed in model studies [Hopfield and Herz, 1995; Tsodyks and Sejnowsky, 1995; van Vreeswijk and Sompolinsky, 1997].

Figure 2.16: Example of pulse codes. Taken from (Gerstner, 1999).

2.10 Local vs. Distributed Codes

[See Churland and Sejnowski (1992)]

Chapter 3

Selectivity and Discrimination

3.1 Quantifying Detection and Discrimination.

The attempt to quantify the accuracy of people's perceptions can be traced back to Gustav Fechner (1801-1887), the father of the field known as **psychophysics**. Fechner's goal was to uncover the natural laws of perception (hence psychophysics) and the first step was to develop quantitative, objective methodologies for measuring the accuracy of perception. To avoid difficulties related to subjective perception, Fechner focused on the use of simple to choice tasks to determine the magnitude of the weakest stimulus that leads to perception – the **absolute threshold** – and the smallest change in stimulus parameters that can be perceived – the **difference threshold** measured in units of **just noticeable difference (JND)**. In attempting to measure these magnitudes, one encounters the fundamental difficulty that presenting identical stimuli doesn't always lead to identical results. All sources of such uncontrolled variability are lumped under the concept of **noise**. In any perceptual experiment, noise can be introduced by the experimental apparatus (the same settings on a stimulus generator don't necessarily lead to the presentation of identical stimuli), variability introduced by the components of the nervous system (so-called neural noise) or changes in the behavioral or attentional state of the subject.

The beginning part of this chapter will present the rudiments of **signal detection theory**. Not only does this serve as the basis for the filed of psychophysics, this methodology (as developed by Fechner and later refined) is one of the main ways of quantifying the fidelity of the neural code. Because the same measures can be applied to neural responses as well as perception, this approach has the advantage of allowing direct comparisons between neural responses and behavior. Moreover, because the method generally sets up a choice between two alternatives, important mathematical concepts can be introduced in their simplest form.

3.2 Detecting a Stimulus

Consider the following experiment designed to measure your detection threshold for light intensity. A tone goes off and you are asked if this tone was followed by a flash of light. The computer generating the stimuli randomly interleaves trials in which a flash is present or not. This procedure is repeated for varying intensities of the light flash. If the experiment is calibrated properly, some of the light flashes are too dim to see, while others are clearly visible. Plotting the percentage of correct guesses as a function of light intensity is known as a **psychometric** function for this experiment (see fig. 3.1). The vertical axis runs from 50% (guessing) to 100% (perfect performance). Because of noise, we don't expect this function to show a perfect cutoff: at some intensities you perform above chance, but not perfectly. **Perceptual threshold** is commonly defined as the intensity level corresponding to 75% correct. Note that the figure of 75% is an arbitrary level and is set by convention.

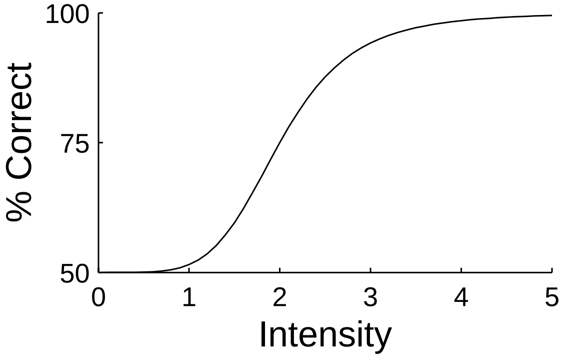


Figure 3.1: Idealized psychometric function in a detection task. Threshold is 2 units in this experiment.

Exactly the same framework can be applied to experiments aimed at determining the discrimination thresholds. One first picks a reference stimulus. On half the trials the reference stimulus is presented and on the other half of the trials, stimuli are presented that differ from this reference stimulus by a fixed amount. In this set up, the presence or absence of a change in the stimulus from the referent is exactly analogous to the presence or absence of a stimulus in a detection task. In the rest of this chapter, we will use language that is applicable for detection experiments, but all the analysis can easily be applied to the case of discrimination experiments.

3.3 Modelling the Detection Task

First we introduce some notation for describing the outcomes of the experiment. On a given trial, there are two possible stimulus configurations: the stimulus can be present (denoted S^+) or absent (denoted S^-). There are also two possible decisions: the subject can respond "yes, the stimulus was present (D^+) or "no, the stimulus was absent" (D^-). This leads to four possible combinations for each trial: a correct "yes" responses on trials where the stimulus was present ("hits"), a correct "no" responses on trials when the stimulus was absent ("correct rejections"), an incorrect "no" response on trials when the stimulus was present ("misses"), and an incorrect "yes" response on trials when the stimulus was absent ("false alarms").

		Response	
		D^+	D^-
Stimulus	S^+	hit	miss
	S^-	false alarm	correct rejection

FOR FIGURES IN THIS SECTION, SEE THE SIGNALDETECT DEMO.

In the standard model of the detection task, it is assumed that sensory processing results in an "internal response" which can be summed up by a single number representing the perceived strength of the stimulus on that trial. Due to the various sources of variability, this response is a random variable, which we denote by R. The dependence of perceived strength on the presence or absence of the stimulus can be represented by plotting the distribution of responses given the stimulus ($\mathcal{P}(R|S^+)$), and the distribution of responses with no stimulus ($\mathcal{P}(R|S^-)$). The distributions of perceived strength for the stimulus trials and no-stimulus trials are often modelled as Gaussian distributions (figure 3.2). Note that while the average perceived strength on trials where no stimulus was shown should be zero, neural noise and noise from the background luminance will give rise to a range of perceived strengths. The decision process is modelled as a simple threshold criterion: if the perceived strength is sufficiently large, subjects respond that they detected a stimulus on that trial. In this figure, the threshold has been set at the crossing point of the two distributions. The fraction of incorrect responses are represented by the area of the shaded areas: the black area represents the fraction of trials where the stimulus was present, but perceived strength was below threshold ("misses") and the light gray area represent trials where the stimulus was absent but perceived strength was above threshold ("false alarms").

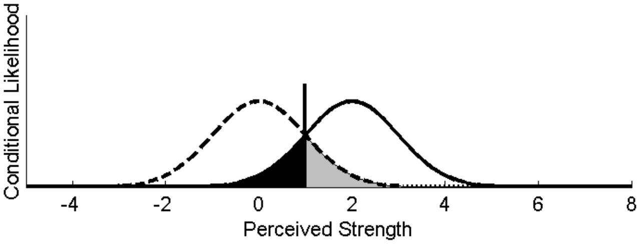


Figure 3.2: Basic model describing a detection task. Solid curve: $\mathcal{P}(R|S^{+})$. Dashed curve: $\mathcal{P}(R|S^{-})$. Black area: misses ($\mathcal{P}(D^{-}|S^{+})$). Gray area: false alarms ($\mathcal{P}(D^{+}|S^{-})$).

3.3.1 Changing Signal Strength

The picture in figure 3.2 represents multiple trials of a detection experiment for a fixed stimulus. As the stimulus is made more salient (*e.g.* a brighter dot in a visual detection experiment), trials in which the stimulus is present will lead to a greater perceived strength on average and the corresponding distribution will move to the right figure **??**A. In this case, only small portions of each distribution lie on the wrong side of threshold, and performance is high. For very dim stimuli, the two distributions will be nearly overlapping and the probability of an error will approach 50% (figure **??**C). Under this framework, varying signal strength will change the location of the perceived signal strength distribution, and the fraction of the two distributions lying on the correct side of threshold will map out the psychometric function (figure **??**D). Note that the perceived signal strength might not be strictly proportional to the actual signal strength. Changing this relationship will change the shape of the psychometric function (*e.g.* **??**E).

3.3.2 Changing the Noise Level

Another way to alter performance in the task is to change the level of noise in the task. For example, one could design an experiment where subjects are asked to detect the presence of a tone with differing levels of background sounds. Increasing noise is modelled as a greater uncertainty in perceived strength, and hence a broadening of the corresponding distributions (figure **??**). Increased noise increases the overlap of the two distributions and hence reduces performance.

3.4 Neurometric Functions

Now consider the same basic experimental set-up, but this time spike trains are being recorded from a neuron in the visual system that responds to light onset. Assuming that the neuron increases its spike rate as stimulus intensity increases, we can can view the number of spikes as a measure of the neurons "perceived signal strength." That is, we can reproduce figure 3.2 by simply replacing the distributions by histograms of the number of spikes per trial. By setting threshold where the two histograms cross, we could use the response of the neuron to guess whether the stimulus was present or absent on a given trial. In this way we can evaluate the ability of the neuron to detect the stimulus. Plotting the percent of correct classifications vs. stimulus strength is known as a **neurometric function**. Using this method, we can compare the behavioral performance of an entire animal on a signal detection task to the ability of individual neurons to perform the task.

In a series of elegant experiments Ken Britten, Bill Newsome and colleagues performed a series of experiments comparing the psychometric functions from monkeys performing a visual motion task with a neurometric function obtained from neurons in an area of the brain specialized for visual motion processing. See figure – WILL DISCUSS IN CLASS.

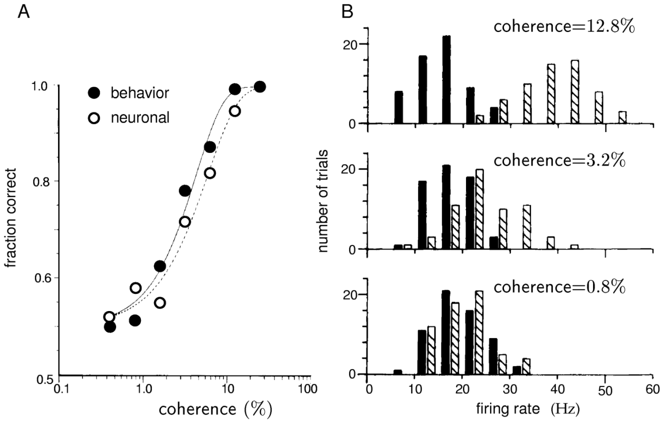


Figure 3.3: A. neurometric and psychometric functions on a visual motion discrimination task. B. Histograms of the number of spikes recorded from a motion sensitive neurons in visual area MT of a monkey. The stippled bars represent trials where the motion in the preferred direction of motion for the neuron. The black bars represent trials where the motion in the anti-preferred or null direction. Coherence relates to the strength of motion signal embedded within a random dot stimulus. Figure from **?**;taken from Dayan and Abbott (2001)

.

3.5 Response Bias and ROC Curves

An important difference between the measurement of a psychometric function and the construction of a neurometric function is that in the behavioral experiment only the yes/no responses are available, whereas in the neural experiment the entire histogram of responses is available. This difference is crucial, since behavioral performance is not only dependent upon the overlap of the two distributions, but also upon the setting of the response threshold. For example, two different subjects may perform differently on a detection task, even if their "internal perception" of the stimulus was identical. For example, performance could be suboptimal for a subject who is hesitant to respond yes unless he or she was pretty confident that a stimulus was actually present.

One way to partially recover the shape of the underlying distributions of perceived strength is to examine both hit rate and correct rejection rate as the threshold is varied rather than just averaging the two to get an overall percent correct. A subject with a strict threshold will have few false alarms, but will have a reduced number of hits. A subject with more lax threshold will have a greater fraction of hits, but at the expense of a greater number of false alarms. A simple way to mimic a variation in decision threshold is to ask subjects to report their degree of confidence in their behavioral choice, say on a scale of 1 to 5 (with 3 meaning very confident in their choice and 1 being nearly a toss-up). A strict threshold can be mimicked by reanalyzing the data where you assign a no response to all trials except where the subject said yes with a high confidence level. A lax threshold would assign a yes response to all trials except confident rejections, etc. In this way, at each level of signal strength the tradeoff between hits and false alarms changes with threshold can be mapped out. We will discuss other ways of varying the decision threshold below.

The function that maps out this tradeoff is known as an **ROC curve** (receiver operating characteristic curve). False alarm rate is represented on the horizontal axis, and hit rate is mapped on the vertical axis. Very high thresholds lead to few false alarms but also few hits, corresponding to the lower left corner of the plot. As threshold is decreased, the number of hits goes up without much increase in the false alarm rate. As threshold is decreased further, the hit rate continues to rise but now the false alarm rate begins to increase. The increasing false alarm rate makes the curve begin to bend to the right. As threshold is reduced even further, the hit rate approaches 100%, but the false alarm rate also increases. At very low thresholds, the subject reports a stimulus on every trial. The hit rate and false alarm rate are both 100%, corresponding to the upper right corner of the plot.

A single ROC curve maps out the tradeoff between hits and false alarms at a single stimulus intensity. Changing the stimulus intensity leads to a new ROC curve. Note that the best performance is represented by points at the upper left hand portion of the ROC plots: few false alarms (left part of plot) and lots of hits (upper part of plot). Therefore, if we make the task easier by increasing the intensity of the stimulus, more of the curve moves to the upper left. If the task is very difficult, hits and false alarms co-vary. A high threshold leads to few hits and few false alarms, a low threshold leads to many hits but at the cost of many false alarms. Therefore, for difficult tasks the ROC curve hugs the diagonal

3.5.1 Neural ROC Curves

When recording neural data, we presumably can construct the relevant response histograms; an ROC curve gives no additional information. But in psychophysical settings, we measure only yes/no responses; the shape of the underlying distributions are inferred. Using an ROC curve to describe neural performance allows for a direct comparison of neural and behavioral data.

3.5.2 Two-Alternative Forced Choice Tasks

An alternative to measuring the effect of varying the threshold criterion is to design experiments that encourage the optimal placement of threshold. On such design is the **two-alternative forced choice task**. The main idea here is to try to make the two response alternatives as similar as possible, thereby greatly reducing opportunities for bias. Generally, each trial contains two "presentations" one where the stimulus is present and the other where the stimulus is absent. Continuing with our visual detection example, one could flash the weak stimulus in one of two locations or flash the stimulus in one of two consecutive stimulus periods. The subject would then be asked to make a decision of which of the two possible presentations actually contained the stimulus. D^1 will denote a decision that first presentation contained the stimulus, and D^2 will denote a decision that stimulus fell on the second presentation. Presumably, subjects make judgements by choosing the trial that led to the largest internal response, rather than comparing

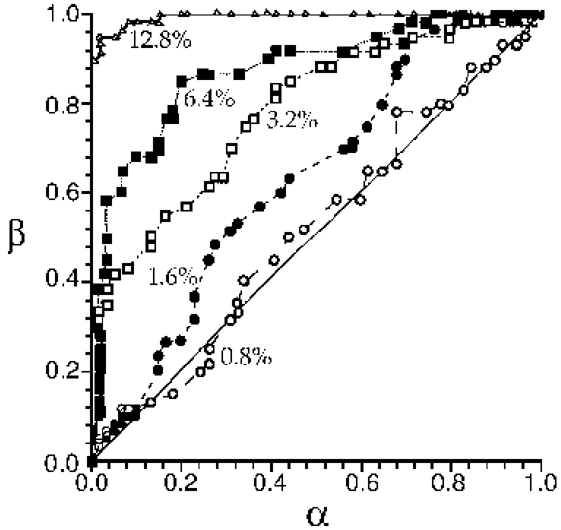


Figure 3.4: ROC curves constructed from spike distributions in a motion discrimination task. Figure from **?**;taken from Dayan and Abbott (2001).

this response to a subjectively determined threshold. Letting R_1 be the random variable describing perceived signal strength (response) on presentation 1 and R_2 describe perceived signal strength on the presentation 2. That is, the decision is made based on whether $R_1 - R_2$ is positive or negative.

We can depict a two-alternative forced choice task by a picture that is very similar to figure 3.2, by calculating the conditional distributions $\mathcal{P}(R_1 - R_2|S^1)$ and $\mathcal{P}(R_1 - R_2|S^2)$. On trials when the stimulus was present on presentation 1, $R_1 = R^+ = R|S^+$ and $R_2 = R^- = R|S^-$. On trials when the stimulus was present on presentation 2, $R_1 = R^- = R|S^-$ and $R_2 = R^+ = R|S^+$. Therefore the two conditional distributions $\mathcal{P}(R_1 - R_2|S^1) = \mathcal{P}(R^+ - R^-)$ and $\mathcal{P}(R_1 - R_2|S^2) = \mathcal{P}(R^- - R^+)$ will simply be negative images of each other. As long as the subjects treat presentation 1 and presentation 2 symmetrically, threshold will be set at zero.

3.6 Response Strategies and Bayes' Rule

The ROC curve can be used to quantify performance for varying thresholds. Now we return to the question of how to set threshold to get optimal performance. In section **??** we implicitly adopted a **maximum likelihood (ML)** strategy, *i.e.* the appropriate yes/no decision on a given trial is given by the stimulus that had the greatest likelihood of generating that internal response, assuming that stimulus was indeed the stimulus presented. In other words, the two curves in the top plot of figure 3.2 represent the two probability distributions $\mathcal{P}(R|S^+)$ and $\mathcal{P}(R|S^-)$. The response was determined by which one of these values was greatest, *i.e.* threshold was placed at the point where the two distributions crossed.

The difficulty with the maximum likelihood strategy is illustrated by the following classic problem.

> Suppose that you go into the doctor's office for a series of tests, and a test for disease X comes up positive. You begin to worry, especially when the doctor says the test is 99% accurate. What's the chance that you actually have disease X? Making the analogy

> with our presentation of maximum likelihood, there are two stimuli (either you have the disease or you don't), and two responses (the test can be positive or negative). Since your test came out positive, according to the maximum likelihood estimator, you should assume that you have disease X since if having the disease gives a positive result 99% of the time, whereas not having the disease gives a false positive only 1% of the time.
>
> Feeling quite concerned at this point you ask the doctor what you should do. She says that you shouldn't worry. Out of the 10 million people that take the test for disease X, only 1 out of 100,000 are likely to have the disease. That means that of the 9,999,900 that don't have the disease, there will be 99,999 false positives, while the 100 people that do have the disease will yield 99 true positives. Therefore, if we consider the total population, the fraction of the time that a positive result actually means that you have the disease is 99/(99+99,999) or less than .1%. You leave the doctor's office quite relieved, but your confidence in the maximum likelihood strategy has been shattered. What happened?

We'll analyze this example in some detail, because it illustrates a number of important points. We first introduce some notation. We have two possible "stimuli": either the patient has the disease (denoted S^+) or they don't (denoted S^-). We also have two possible responses: the test either comes up positive (R^+) or the test is negative (R^-). In the maximum likelihood strategy, if we are forced to choose which stimulus gave rise to a certain response r, we choose the stimulus that maximizes $\mathcal{P}(r|s)$, *i.e.* we choose the stimulus that would have the greatest likelihood of generating that response conditioned on that stimulus was the one actually presented. But what we'd really like to choose is the stimulus that maximizes $\mathcal{P}(s|r)$. The strategy of choosing the stimulus that maximizes $\mathcal{P}(s|r)$ is known as the **maximum *a posteriori*[1] (MAP)** estimate. In our disease example, what we know is $\mathcal{P}(r|s)$ for various combinations of r and s – $\mathcal{P}(R^+|S^+) = \mathcal{P}(R^-|S^-) = .99$ and $\mathcal{P}(R^-|S^+) = \mathcal{P}(R^+|S^-) = .01$ – while we'd like to know $\mathcal{P}(S^+|R^+)$.

The rule for combining these conditional probabilities is known as Bayes' rule. To illustrate this rule, first consider the two dimensional histogram of the results of applying the test to 10 million people (fig. **??**). If we divide the number of cases in each bin by 10 million, we obtain the joint probability distribution of the disease variable d (equal to S^+ or S^-) and the test variable t (equal to R^+ or R^-). To calculate the number of people that both have the disease and test positive (99) one can calculate that 100 people have the disease overall (10 million $\times 1/100,000$), and of these 100 people 99 will test positive. In probabilistic terms,

$$\mathcal{P}(S^+, R^+) = \mathcal{P}(S^+)\mathcal{P}(R^+|S^+) \tag{3.1}$$

Geometrically, $\mathcal{P}(S^+)$ captures the portion of the total number of people in the S^+ row of the distribution, while $\mathcal{P}(R^+|S^+)$ captures the portion of this row that tests positive. Alternatively, one could find the total portion $\mathcal{P}(R^+)$ of the population that tests positive, *i.e.* the portion of the distribution in the R^+ column, and multiply this by the fraction $\mathcal{P}(S^+|R^+)$ that actually had the disease. While the problem was defined in terms of the probabilities in equation (3.1), it is nevertheless true that this alternative strategy leads to the right answer, *i.e.*

$$\mathcal{P}(S^+, R^+) = \mathcal{P}(R^+)\mathcal{P}(S^+|R^+) \tag{3.2}$$

Combining these two equations we find that

$$\mathcal{P}(S^+|R^+) = \frac{\mathcal{P}(S^+)\mathcal{P}(R^+|S^+)}{\mathcal{P}(R^+)} \tag{3.3}$$

---

[1] "After the fact."

Similarly

$$\mathcal{P}(S^-|R^+) = \frac{\mathcal{P}(S^-)\mathcal{P}(R^+|S^-)}{\mathcal{P}(R^+)} \tag{3.4}$$

Note that evaluating what to do about your positive test result, you don't really care about the overall probability that you tested positive, you already know that. What you are interested in is the relative likelihoods of the various possibilities *given* that you tested positive. Therefore you don't really care about the denominator in equation (3.3). What you care about is the **likelihoods** $\mathcal{P}(R^+|S^+)$ and $\mathcal{P}(R^+|S^-)$ and your **prior** knowledge about the prevalence of the disease $\mathcal{P}(d)$. In words, Bayes' rule says that the **posterior** probability ($\mathcal{P}(S^+|R^+)$) is proportional to the likelihood ($\mathcal{P}(R^+|S^+)$) times the prior $\mathcal{P}(S^+)$.

3.7 Forward and Reverse Perspectives

SECTION NEEDS TO BE REWORKED

Notice that the difference between ML and MAP strategies has to do with your prior knowledge. If you have no prior reason to believe that one stimulus is more likely than another, then the ML and MAP strategies arrive at the same answer. Therefore, for the detection tasks described above, as long as the flash is presented on half the trials, then the stimulus and no stimulus conditions have equal prior probabilities. Hence analysis presented above apply to both ML and MAP frameworks.

Consideration of prior probabilities can play an important role in terms of interpreting physiological results. In particular, the nervous system has presumably been shaped by evolution to give optimal (or nearly optimal) performance *given the probabilities of stimuli encountered by the animal during natural behavior.* Thus if the performance of the nervous system (or the animal) is below optimal for some set of stimuli, it may simply be that the stimuli chosen by the experimenter are not presented with the same relative probabilities encountered in natural settings. For this reason, recently there has been growing interest in trying to obtain quantitative estimates of the statistics of natural scenes.

3.7.1 Optimizing Pay-off

One could argue that to survive it may be more important to be right about some things than others. For example, it might have been useful for our ancestors to detect the rustling of prey in some underbrush, and imperative to recognize whether the rustling was caused by a tiger. Note that it is relatively easy to incorporate such different payoffs within a probabilistic framework to determine optimal behavioral strategies. Within our two choice framework, we can assign payoffs and penalties for each of the possibilities. For example, correctly guessing the presence of a stimulus might lead to 2 units of reward, while a false alarm leads to a penalty of 5 units. A missed stimulus may have a penalty of 1 unit and a correct guess of a non-stimulus trial might lead to a 1 unit reward. This situation can be summarized in the following payoff matrix:

		Response	
		D^+	D^-
Stimulus	S^+	2	-1
	S^-	-5	1

It important to note that as long as each stimulus is drawn independently, the best that can be done is to find the optimal course of action for each response level. So for now we will fix a response level $R = r$ and calculate the probability that a stimulus was present or absent give r, *i.e.* we need to calculate $\mathcal{P}(S^+|r)$ and $\mathcal{P}(S^-|r)$. If we say yes (S^+) when experiencing a response level

r then the average reward for this condition will be $2\mathcal{P}(S^+|r) - 5\mathcal{P}(S^-|r)$ while if we say no (S^+) the average reward will be $-\mathcal{P}(S^+|r) + \mathcal{P}(S^-|r)$. Therefore we should guess yes for all responses where

$$2\mathcal{P}(S^+|r) - 5\mathcal{P}(S^-|r) > -\mathcal{P}(S^+|r) + \mathcal{P}(S^-|r) \tag{3.5}$$

A little algebra reveals that the condition for a yes response is

$$\mathcal{P}(S^+|r)/\mathcal{P}(S^-|r) > 2 \tag{3.6}$$

That is, if a given response level leads us to believe that it is twice as likely for a stimulus to be present than not, we should guess yes. Otherwise we should guess no. This conservative strategy is dictated by the relatively high penalty for false alarms.

Problems

Chapter 4

Signal, Noise and Information

4.1 Continuous Stimuli

In the last chapter we focused on experiments where the number of stimuli is small, as in two alternative forced choice tests. However, it is often the case that stimuli are organized so that there is some notion of distance between stimuli. On the response side, spike rate (or spike number) inherently contains a notion of distance. In fact, in almost all cases tuning curves and response functions are described as continuous functions of continuous variable. In this chapter, we'll focus on cases where the stimulus is described by a single variable, for example the frequency of a tone presented to the auditory system. Tuning curves or response functions describe changes in the average response for different stimuli. But of course, responses are somewhat variable. This chapter describes various methods for quantifying this variability.

> Biological Aside. Most experimentally derived tuning curves show **error bars** indicating the magnitude of the **standard error of the mean (SEM).** While the SEM is *related* to the variability in the response, it does not directly quantify response variability. In particular, repeating the experiment will reduce the size of the error bars, since one becomes more confident in the measurements of the mean response for each stimulus. But running more experiments doesn't reduce the variability in the responses!

4.2 The Signal-to-Noise Ratio

The most direct way of quantifying the level of noise is to measure the variance of the response distribution, averaged over all stimuli. (Recall that the variance is simply the average of the squared distance from the mean - see prob **??**. Taking the square root of the variance yields the **standard deviation**, of denoted by σ, which has the same units as the response variable.) But simply determining that the standard deviation of the response rate is 5 Hz has very little meaning. Should 5 Hz be thought of as a large or a small quantity? To get a handle on this question, one might compare 5 Hz to the typical spike rate of a given neuron. 5 Hz might represent a high level of noise for a neuron that typical spikes at 5-10 Hz, but a relatively modest level of noise for a neuron spiking at 100 Hz. But as we learned in section **??** it the level of *change* in response that is useful for discriminating between stimuli. If changing the stimulus only changed the mean firing rate by a few Hz, then 5 Hz noise would lead to very poor stimulus discrimination, even if typical spike rates were near 100 Hz. For this reason, one often describes the variability in responses using the **signal-to-noise (SN) ratio**, which is simply the variance of the signal divided by the variance of the noise. If noise levels are vary low, then small changes in the denominator can make a large difference in the SN ratio. For this reason, signal and noise are sometimes quantified by calculating the fraction of the total variance in the response that is accounted for by the response function (the gives the mean response level for each stimulus). Since variances are additive, the total variance is simply the variance of the noise plus the variance of the signal. Therefore, the fraction of the variance accounted for by the signal is given by

$$\frac{\sigma^2_{signal}}{\sigma^2_{total}} = \frac{\sigma^2_{signal}}{\sigma^2_{signal} + \sigma^2_{noise}} \tag{4.1}$$

4.3 Signal Estimation

Generally, the signal-to-noise ratio is described on the output end, *i.e.* noise represents the variability in responses for a range of stimuli. However, as we saw in the last chapter, from an animal's perspective the task is to take some internal response pattern and estimate which stimulus was out there in the world. The interesting object here is not the distribution of responses given a stimulus ($p(r|s)$, but the distribution of signals given the response ($p(s|r)$). Given this distribution, there are two basic strategies for estimating the stimulus. First, one can choose the stimulus that is most likely, *i.e.* the stimulus s that gives the maximum value for $p(s|r)$. This is the MAP strategy outlined in the previous chapter (equivalent to maximum likelihood if all stimuli have equal prior probabilities.) Alternatively, one can produce an estimated stimulus s^{est} that is as close as possible to the true stimulus s. To define "as close as possible" we must provide some notion of distance. One way to do this is to define a **loss function**, *i.e.* a function that determines the "cost" of various degrees of error. The most common loss function is $(s - s^{est})^2$. In this case the task is to minimize the squared distance between the estimated and true stimulus. *In this case, the optimal strategy is to set the estimated stimulus for a given response r to be the average of the stimulus distribution conditioned on the response r* (see prob 4.3.1).

Given any mapping from response to stimulus, we can talk about the estimated stimulus s^{est} much as we did the "signal." Then the and the error - the difference between s^{est} and the actual stimulus - as we did the noise. MORE.

Problems

Problem 4.3.1 Show that the mean value is the value that minimizes the least squared error, *i.e.* given a distribution of values $\{x_1, x_2, \ldots, \mathbf{x}_N\}$, the value of y that minimizes $\sum_i (x_i - y)^2$ is the mean value $\bar{x} = \frac{1}{N} \sum_i x_i$.

4.4 Information as Reduction in Uncertainty

In the rest of the chapter we will deal with a topic known as **information theory**, first developed by Claude Shannon and colleagues at Bell Labs in the 40's and 50's. While Shannon introduced the subject as a theoretical framework for studying coding along a communication channel, the introduction here will focus on the concept of **mutual information** as generalizing the signal-to-noise ratio. To make this connection, we first view the variance as a measure of uncertainty. For example, the variance of the noise for a given stimulus represents how uncertain we are about the response to that stimulus. The second connection relates to the formula

$$\sigma^2_{total} = \sigma^2_{signal} + \sigma^2_{noise} \tag{4.2}$$

or

$$\sigma^2_{signal} = \sigma^2_{total} - \sigma^2_{noise} \tag{4.3}$$

σ^2_{total} is the variance of the distribution of responses collected over the entire experiment. It represents the level of uncertainty about the response over the whole range of stimuli, *i.e.* it represents how uncertain one would be about the response if one knew only which *set* of stimuli were being presented, but didn't know the specific stimulus presented. In this formulation, the strength of the signal relative to the noise corresponds to how much knowing which stimulus was presented reduces your uncertainty about the response, *relative to the initial level of uncertainty.* It is in this sense that knowing which stimulus gives you information about the response - it reduces the level of uncertainty. This particular viewpoint means that information is defined as loss in uncertainty. Thus information and uncertainty are flip-sides of the same thing and hence will have the same units.

4.5 Entropy

The key to information theory is to make formal mathematical definition that captures the notion of uncertainty. Suppose that we want to define the amount of uncertainty contained in a **discrete random variable** X. What we mean by X being a discrete random variable is that X has a finite number of states, which we denote by $\{x_1, x_2, \ldots, x_N\}$. X is governed by a discrete probability distribution, that is an assignment of a probability $P(x_i)$ to each state x_i. Remember that the total probability must add to 1: $\sum_i P(x_i) = 1$. We will use $H(X)$ to denote the uncertainty embodied in the distribution X. Shannon outlined three properties that should be satisfied by the function H.

1. 0 uncertainty corresponds to the case where one state has probability 1, and all others have probability 0. In math terms, $H(X) = 0$ if and only if $P(x_i) = 1$ for some x_i.

2. Maximum uncertainty is attained when all states are equally probable, *i.e.* $H(X)$ is maximal when $P(x_i) = 1/N$.

3. The uncertainty contained in a distribution composed of two independent sources of uncertainty should be the sum of uncertainties for these two sources. To make this formal, suppose we have a second (independent) distribution Y composed of states $\{y_1, y_2, \ldots, y_M\}$. Consider the joint distribution (X, Y) with states composed of all pairs (x_i, y_k). The condition we want is $H(X, Y) = H(X) + H(Y)$

Shannon then proved that the only function that satisfies these three properties must be proportional to the function

$$H(X) = -\sum_i P(x_i) \log_2(P(x_i)) \tag{4.4}$$

The uncertainty measure H is known as the **entropy**, since the formula for uncertainty turns out to be the same as that for the older concept of entropy used in statistical physics to quantify the amount of "disorder" in a system of interacting particles. The appearance of a logarithm in the definition of entropy stems from condition 3 and the fact that logarithms convert products into sums ($log(ab) = \log(a) + \log(b)$). In particular, while probabilities for independent events *multiply*, $P((x_i, y_k) = P(x_i)P(y_k)$, the resulting uncertainties must add, $H(X, Y) = H(X) + H(Y)$ (see problem 4.5.1). The choice to use base 2 for the logarithm is by convention and means that uncertainty (and hence information) is expressed in "bits," with the flip of an unbiased coin having one bit of uncertainty since it represents one random binary choice.

4.5.1 Intuitions

There are a number of ways of thinking about the entropy. For example, $H(X)$ can be thought of as the average number of yes no questions it takes to guess which state was chosen at random from the distribution X. Let's consider a particularly simple example. Suppose X has three states with $P(x_1) = 1/2$ and $P(x_2) = P(x_3) = 1/4$. Then the optimal guessing strategy will be to first ask the question, "is the state x_1?" If the answer is yes you are done, and if no the question "is the state x_2?" will determine whether the state is x_2 or x_3. In this example, guessing the state requires one question half of the time and two questions the other half. Thus the average number of guesses, $H(X) = 1.5$. This can be confirmed by plugging the probabilities into the formula:

$$H(X) = -(.5)\log_2(.5) - (.25)\log_2(.25) - (.25)\log_2(.25) = .5 + .25 * 2 + .25 * 2 = 1.5 \tag{4.5}$$

A second way to think of $H(X)$ is to form long "strings" of repeated samples from X, *e.g.* the 10 state string $x_1x_3x_3x_1x_2x_1x_1x_1x_3x_2$. Using the same distribution X as above, we'd expect that roughly half of the entries equal to x_1 and roughly one quarter equal to x_2 and another quarter equal to x_3. If we choose strings with a large number of entries, the chance of getting a string where the state *don't* show up with these relative probabilities will be very small. Since there are three states, the total number of strings of length N is 3^N. One of the results that Shannon proved is that for long strings the number of *typical* strings of length N is equal to $2^{NH(X)}$, with the chance of finding *any* of the other strings being negligibly small.

Another way to think about the number of typical strings is in terms of compression algorithms. If there are a total of 3^N strings, but only $2^{NH(X)}$ are "typical" then one should be able to make up new (shorter) symbol strings for the typical strings, and only occasionally use up a lot of symbols broadcasting the highly unlikely "atypical" strings. If there are a lot of typical strings (*i.e.* $H(X)$ is large) then one will have to come with a lot of abbreviated symbol strings and hence these abbreviations will save less coding space. Thus $H(X)$ is related to the optimal amount of compression that one could achieve. Note that this important result quantifies the maximal amount of compression that *could be* achieved, but doesn't say anything about the nature of the compression algorithm that would achieve that goal. However, if one constructs a compression algorithm, the entropy can be used to determine how close this strategy is to the optimal strategy possible.

4.5.2 Continuous Distributions

So far we have defined entropy for discrete distributions. The most natural way to extend the definition of entropy to distributions over a continuous parameter is to divide the parameter dimensions into discrete bins and calculate the resulting discrete entropy. Then we can define the entropy of the continuous variable as the limit of the entropies as the bin size gets very small. Suppose we parameterize the distribution by the continuous variable x, letting $p(x)$ denote the corresponding probability density function. We then chunk x into bins of width Δx. Since the bin containing x has probability approximately equal to $\Delta x\ \mathcal{P}(x)$, the entropy at this resolution is given by

$$\begin{aligned} H(X) &= -\sum_x \Delta x\ \mathcal{P}(x)\log_2(\Delta x\ \mathcal{P}(x)) && (4.6)\\ &= -\sum_x \Delta x\ \mathcal{P}(x)\log_2(p(x)) - \log_2(\Delta x)\sum_x \Delta x\ \mathcal{P}(x) && (4.7)\\ &= -\sum_x \Delta x\ \mathcal{P}(x)\log_2(p(x)) - \log_2(\Delta x) && (4.8)\end{aligned}$$

In the limit where $\Delta x \to 0$, the first term becomes the integral

$$-\int dxp(x)\log_2(p(x)) \tag{4.9}$$

This is often known as the **differential entropy**. However, the second term $-\log_2(\Delta x) \to \infty$. This captures the notion that with an infinite number of states, the entropy grows infinitely large. But $-\log_2(\Delta x)$ depends only on the resolution, not on the shape of the distribution. Therefore, if we are interested in the difference in entropy between two distributions measured at the same resolution, we can focus our attention on the differential entropy without having to worry about infinite quantities.

Problems

Problem 4.5.1 Show that if X and Y are independent random variables, then $H(X,Y) = H(X) + H(Y)$.

4.6 Mutual Information

Now that we have the proper notion of uncertainty, we return to the problem of quantifying signal and noise. Remember that want to define the information about the response that is gained by specifying the stimulus as the reduction in the uncertainty contained in the entire distribution of responses $P(r)$ to the average uncertainty in the noise. The total uncertainty is given by

$$H(R) = -\sum_r P(r)\log_2(P(r)) \tag{4.10}$$

The noise uncertainty for a particular stimulus s is the uncertainty in the conditional distribution $P(r|s)$:

$$H(R|s) = -\sum_r P(r|s)\log_2(P(r|s)) \tag{4.11}$$

To obtain the average uncertainty across all stimuli, $H(R|S)$, we take the weighted average of the individual uncertainties

$$H(R|S) = -\sum_s P(s)\sum_r P(r|s)\log_2(P(r|s)) \tag{4.12}$$

We use $I(S,R)$ to denote the **mutual information** (sometimes called the **transinformation**) between stimulus and response. This is the analogue of the notion of "signal." $I(S,R)$ is just the difference between the total entropy $H(R)$ and the average noise entropy $H(R|S)$

$$I(S,R) = H(R) - H(R|S) \tag{4.13}$$

Mathematical Derivation. Using little algebra, along with Bayes' rule, we can write the formula for the mutual information in a nice form. The derivation will rely on two substitutions:

$$
\begin{aligned}
P(r) &= \sum_s P(s)P(r|s) && (4.14)\\
P(s,r) &= P(s)P(r|s) && (4.15)
\end{aligned}
$$

So

$$
\begin{aligned}
I(S,R) &= H(R) - H(R|S) && (4.16)\\
&= -\sum_r P(r)\log_2(P(r)) - \sum_s P(s)\sum_r P(r|s)\log_2(P(r|s)) && (4.17)\\
&= -\sum_r \left(\sum_s P(s)P(r|s)\right)\log_2(P(r)) - \sum_{s,r} P(s)P(r|s)\log_2(P(r|s)) && (4.18)\\
&= -\sum_{s,r} P(s)P(r|s)\left(\log_2(P(r)) - \log_2(P(r|s))\right) && (4.19)\\
&= -\sum_{s,r} P(s,r)\left(\log_2(\frac{P(r)}{P(r|s)})\right) && (4.20)\\
&= -\sum_{s,r} P(s,r)\left(\log_2(\frac{P(r)P(s)}{P(s)P(r|s)})\right) && (4.21)\\
&= -\sum_{s,r} P(s,r)\left(\log_2(\frac{P(r)P(s)}{P(s,r)})\right) && (4.22)
\end{aligned}
$$

The most important aspect of equation (4.22) is that it is symmetric in r and s, *i.e.* switching r and s leads to the same formula. This means that mutual information is truly mutual: the average amount of information gained about the response by specifying the stimulus is equal to the average amount of information gained about the stimulus by specifying the response.

$$I(S,R) = H(R) - H(R|S) = H(S) - H(S|R) \qquad (4.23)$$

In other words, looking at the problem from the experimenter's perspective (what response do I get from presenting a give stimulus) and the organism's perspective (what stimulus gave rise to a given internal response) gives the same answer.

4.7 Maximizing Information Transfer

Equation (4.23) has a number of important consequences. For example, given a fixed amount of noise entropy $H(R|S)$, the coding strategy that maximizes the mutual information between stimulus and response will lead to a distribution of responses that maximizes $H(R)$. What distribution of responses give the maximal entropy? This question only makes sense of there is something constraining the possible range of responses - an unlimited range would lead to entropy values that approach infinity. Thus maximizing entropy (and hence information transfer) must be done in the context of some constraint.

For example, suppose that we use firing rate as our measure of response and suppose that there is some maximum firing rate. Then the distribution of responses that maximizes entropy and hence mutual information is one where all firing rates between zero and the maximum are equally likely. As an example of this, consider the case of responses of a contrast-sensitive neuron in the fly visual system known as LMC. Figure 4.1) shows a plot of contrast vs. response (points) and a theoretically derived curve that represents the integral of the distribution of the contrasts encountered in the fly's natural world. One can see that there are relatively few stimuli at extreme contrasts (near ±1). Thus to contrast must change quite a bit before the response changes significantly. Near 0 contrast, a large number of stimuli fall within each relatively small range of contrasts, so responses change more rapidly.

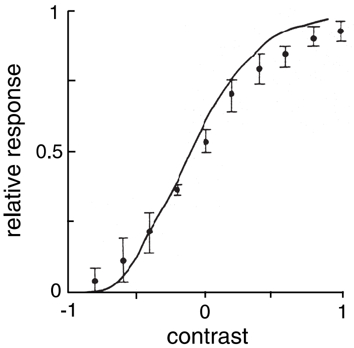


Figure 4.2: Contrast response of the fly LMC (data points) compared to the integral of the natural contrast probability distribution (solid curve). The response is the amplitude of the membrane potential fluctuation produced by the onset of a light or dark image with a given level of contrast. Contrast is defined relative to the level of illumination that produced a half-maximal response, $\bar{I}$. For a light intensity I, the contrast is defined as $(I - \bar{I})/\bar{I}$. The relative response is the actual response divided by the maximum response for a contrast of one. (Adapted from Laughlin, 1981)

Figure 4.1: Adapted from Dayan and Abbott, 2001

If one fixes the *mean* response rather than its maximum, what is the distribution of responses? One can show that in this case the distribution of firing rates should be exponential. Data recorded from a number of visually responsive areas of the cortex of monkeys that are watching TV show a roughly exponential distribution (Rolls et al.). The distribution that maximizes information given a fixed variance is a Gaussian. The differential entropy of a Gaussian with standard deviation σ is $\log_2(\sigma\sqrt{2\pi e})$. MORE.

4.8 Measuring Mutual Information

There are a number of approaches to measuring mutual information. First is the so-called direct method. In this approach, one measures the response entropy and noise entropy by gathering enough data to estimate the required distributions and calculate their entropy. This approach has been pursued in only a few cases since gathering enough data is often not feasible, unless one wants to make strong simplifying assumptions. For example, suppose one doesn't want to commit oneself to a rate coding *assumption* from the outset. Then each spike train must be treated as a separate response. Even if only brief responses are considered, the number of possible spike trains becomes staggering. Considering a brief response period of 100 msec and characterizing the resulting spike trains with a resolution of 10 msec, leads to at least $2^1$0 possible spike trains, without even considering cases where two spikes fall within a single 10 msec bin. However, certain short cuts and estimates can be taken and this method has been successful in a number of cases.

4.8.1 A Lower Bound on Mutual Information

Much more common is to use short-cuts and assumptions to bound the real mutual information within some range. In the most common approach focuses on entropy calculations in terms of the stimulus:

$$I(S,R) = H(S) - H(S|R) \tag{4.24}$$

Since the stimulus distribution is decided upon by the experimenter, $H(S)$ is usually known. Therefore, if we can get an *upper* bound on $H(S|R)$, we can get a *lower* bound on the mutual information $I(S,R)$. Getting a true measure of $H(S|R)$ may require a lot of data, particularly if responses are characterized in a complex manner. However, suppose the stimulus is parameterized by a single parameter (for example velocity) and we have some method of forming an estimate of the stimulus s^{est}. Since the variance of the distribution $(S|r)$ is the minimum squared error for any estimator, $\langle (s - s^{est})^2 \rangle$ gives an upper bound on the variance. Since a Gaussian distribution is the distribution that maximizes the entropy for a given variance, we can say that

$$I(S,R) \geq H(S) - \log_2(\sqrt{\langle (s - s^{est})^2 \rangle \, 2\pi e}) \tag{4.25}$$

4.8.2 An Upper Bound on Mutual Information

The most common way to get an upper bound on the mutual information is to calculate the theoretical maximum for the entropy in a spike train. This will give an upper bound on $H(R)$. Then, if we simply ignore the noise entropy $H(R|S)$, then this upper bound will also give a (rather loose) upper bound on the mutual information since

$$I(S,R) = H(R) - H(R|S) \tag{4.26}$$

If spikes are measured with infinite resolution, then the maximum entropy would be infinite. But suppose we put spikes into bins of size Δt and choose small enough bins so that there is at most one spike per bin. If responses have length T then there are $T/\delta t$ bins, and if we assume that there is the mean number of spikes equal to R over this period, then the maximal entropy is obtained when each bin has the same probability of seeing a spike, $R/(T/\delta t)$. The resulting entropy is the maximal entropy of any distribution of spike train responses.

4.9 Using Information as a Relative Measure

4.10 Measuring the Role of Correlation in Neural Coding

How much do correlations between spikes contribute to coding? Note that the notion of correlation includes issues of both temporal and population encoding. One way to define **temporal coding** is a code in which the contribution of one spike in a spike train depends on the existence and location of other spikes in the trains. Otherwise, one could consider this as a rate modulated code. (Note that this is a different definition of temporal coding than Theunissen and Miller. How?) Similarly, one can define a **population code** as a code where the contribution of one neuron to the decoding of the stimulus depends on the response of other neurons. From this perspective, the key issue is to determine whether correlations between spikes contribute to coding. Several information-based measures of correlation coding have been proposed based on this general idea.

Here I will present a systematic breakdown of these measures with the aim of honing the underlying intuition behind each approach. All calculations will be based on different decompositions of stimulus and response distributions. To measure correlation we will assume that a response r can be decomposed into a list of individual responses r_i, $i = 1, \ldots, N$. In a population coding setting, r_i might represent the firing rate of neuron i. For temporal coding, the response period could be broken into N time bins and r_i would represent the number of spikes during that time bin. If we had more data, we might consider the possibility of "spatio-temporal" encoding in which we consider responses in multiple time bins from multiple neurons. Finally, while spike counts/rates are by far the most common representation of spike trains, the r_i could be based on any convenient encoding of the response. For example, r_i might represent the latency to response of the ith neuron.

4.11 Three Ways of Ignoring Correlations

We would like to quantify how much correlations between responses contributes to neural coding. The approach we will take is to assume that we are given a information about individual responses but not their correlations. From that limited information we then make an estimate of the mutual information under the default assumption that correlations do not play a role. We can use the difference between this estimated value for mutual information to the true value of the information as a quantitative estimate of how much correlations contribute to the code. For convenience, we will use population coding language in which the variable r_i represents the response of the ith neuron.

In the first example of this approach, assume that someone gives you the mutual information between each the set of responses R_i and the stimulus set, $I(S, R_i)$. If the sets of responses were mutually independent, then we know that the information about the stimulus contained in the set of all responses would simply be the sum of the separate values for the mutual information, *i.e.* $\mathbf{I}_{sep}(S, R) = \sum_i I(S, R_i)$. If $\mathbf{I}_{sep}(S, R)$ is smaller than the true value for the mutual information $\mathbf{I}(S, R)$, this indicates that responses are **redundant**, *i.e.* some of the information contributed by an individual response distribution r_i can be extracted from knowing the other responses. On the other hand, if $\mathbf{I}_{sep}(S, R) > \mathbf{I}(S, R)$, this indicates that responses are **synergistic**, *i.e.* we get *more* information from looking at the pattern of responses than simply looking at responses individually. Seeing the world as half-full rather than half-empty, researchers have used the quantity $\Delta I_{synergy} = \mathbf{I}_{sep}(S, R) - I(S, R)$ as a measure of the contributions of correlations to neural coding. (Negative values for $\Delta I_{synergy}$ indicate redundant coding.) For an example, see figure XX below (TBA).

Now suppose that you were given the stimulus and response data for the individual neurons, but you knew nothing about the correlation between neurons per trial. This is actually a common situation that experimenters find themselves in, simply due to technical considerations. While simultaneous recordings from multiple neurons are becoming more common, it is still common in many experiments to position an recording electrode at a single position, record responses from a single neuron to a battery of stimuli, then move the electrode forward and repeat the stimulus set while recording from a different neuron. In cases where you did have simultaneous recordings from multiple neurons, you can artificially recreate this situation by "shuffling" the responses, /ie/ if one records responses to 50 presentations of stimulus s, one can reconstruct a typical population response to s by combining the responses from different trials across neurons. For example, if you have just two neurons a typical response pair might look like $\left[r_1^1 3(s), r_2^6(s)\right]$, where $r_i^n(s)$ is the response of the ith neuron to the nth presentation of stimulus s. This process relies on the response properties of each neuron, but since the actual recordings are obtained from different presentations of the stimulus, the process eliminates the use of any possible cues coming from the correlations between neurons on a single trial to decode the stimulus - exactly what we were looking for.

Based on the set of these **shuffled** stimlus/response pairs, we can generate a new joint probability distribution between stimulus and response defined by having independent response distributions for each stimulus, *i.e.* $P_{sh}(r|s) = \prod_i P(r_i|s)$. We can use this new "shuffled" distribution to calculate an estimated value for mutual information, $I_{sh}(S,R)$, that ignores correlations between responses across neurons for each stimulus. The difference between $I_{sh}(S,R)$ and the true information is called $\Delta I_{sh} = I_{sh}(S,R) - I(S,R)$. Like $\Delta I_{synergy}$, ΔI_{sh} can be positive or negative.

The third approach is related to the second, but relies on an interpretation of mutual information $I(S,R)$ as measuring the optimal performance of a decoding algorithm averaged over all (s,r) pairs. (This is equivalent to the optimal performance of an algorithm trying to predict responses give a the stimulus.) So suppose you were given the shuffled (or separately recorded) data, and you developed an optimal decoding algorithm based on the shuffled distribution. Now, you measure the average performance of your algorithm, but this averaging takes place over the *actual* distribution of stimulus/response pairs. The difference between this level of performance and the true optimal performance given by the mutual information $I(S,R)$ has been denoted ΔI. It is easy to see from this presentation that ΔI is always less than equal to zero, since it evaluates the performance of an algorithm optimized for the shuffled distribution as applied to the true distribution.

4.12 A Systematic Decomposition

We have three information measures at play here: the sum of the separate measures of mutual information, $I_{sep}(S,R) = \sum_i I(S,R_i)$, the mutual information between the stimulus and the shuffled responses, $I_{sh}(S,R)$, and the true mutual information, $I(S,R)$. What we will show is that we can start with the separate measure of information, and add a number of correction terms to get to the true mutual information, passing through the shuffled information along the way. In doing so, rearranging terms we will construct a decomposition of the true mutual information into parts that represent different contributions of correlated firing. For the first decomposition we write

$$I = I_{sep} + (I_{sh} - I_{sep}) + (I - I_{sh})$$

This can be directly compared to the decomposition proposed by Panzeri and colleagues,

$$I = I_{lin} + I_{sig-sim} + I_{corr-ind} + I_{corr-dep}$$

with $I_{lin} = I_{sep}$ described as the "linear" term, $I_{sig-sim} = I_{sh} - I_{sep}$ the "signal similarity" term, and $I_{corr-ind} + I_{corr-dep} = I - I_{sh}$ is the sum of the "correlation independent" and "correlation dependent" noise terms. The correlation dependent noise term $I_{corr-dep} = \Delta I$ described above. Formulas for these terms are derived below.

To start the analysis, let's look at I_{sep}:

$$I_{sep}(S,R) = \sum_i I(S,R_i) = \sum_i H(R_i) - \sum_i H(R_i|S) = \sum_i H(R_i) - \sum_i \sum_s H(R_i|s)$$

The first term is the sum of the entropy of the total response distributions R_i. The second term is the sum of the entropies of the response distributions conditioned on the stimulus: the "noise entropies." In calculating I_{sep}, we are implicitly assuming independence between the total response distribution across cells, as well as independence of the noise distribution across cells.

In contrast, I_{sh} is calculated by constructing a distribution where independence between cells is assumed for the noise (the variability of responses remaining after specifying the stimulus), and the total distribution is constructed from there. So we have

$$\begin{aligned} P_{sh}(r|s) &= \prod_i P(r_i|s) \\ P_{sh}(r) &= \sum_s P(s) P_{sh}(r|s) \end{aligned}$$

Therefore, the noise term will be exactly the same for I_{sep} and I_{sh}, so that

$$\begin{aligned} I_{sig-sim} &= I_{sh}(S,R) - I_{sep}(S,R) \\ &= \left(H_{sh}(R) - \sum_i H(R_i)\right) - \left(H_{sh}(R|S) - \sum_i H(R_i|S)\right) \\ &= \left(H_{sh}(R) - \sum_i H(R_i)\right) \end{aligned}$$

Since the sum $\sum_i H(R_i)$ is equal to the joint entropy when responses are independent, this term captures the effects of dependencies of total response distributions across neurons, ignoring the stimulus. This is why Panzeri et al. named this the "signal similarity" term. Since entropy is maximal when responses are independent, this term is always negative. Note that

$$\Delta I_{sh} = I_{sh} - I = (I_{sh} - I_{sep}) + (I_{sep} - I) = I_{sig-sim} + \Delta I_{synergy}$$

Since $I_{sig-sim} \leq 0$, this implies that $\Delta I_{synergy} \leq \Delta I_{sh}$.

The easiest example for thinking about this term is the case of recording from neurons with overlapping tuning curves. If one neuron responds to a stimulus, then you can assume that the other neuron is more likely than average to also respond, even without knowing anything about the stimulus. This signal similarity (redundancy) serves to reduce the amount of mutual information between the response pairs and the stimulus.

Now we need to understand $\Delta I_{sh} = I - I_{sh} = I_{corr-ind} + I_{corr-dep}$. Before we begin, let's write

$$I(S,R) = H(S) - H(S|R) = -\sum_s P(s)\log(P(s)) + \sum_r P(r) \sum_s P(s|r)\log(P(s|r))$$

Remember that we can interpret $-\log(P(s))$ as the number of yes/no questions it would take to guess that a randomly chosen stimulus was equal to s. Then the entropy $H(S) = -\sum_s P(s)\log(P(s))$ is the average number of guesses we need averaged over the whole stimulus set. Similarly, $\log(P(s|r))$ is the number of guesses that we need to determine that the stimulus was s given that we knew the response was r.

Now we can write

$$\Delta I_{sh} = I - I_{sh} \quad = \quad H(S) - H(S|R) - (H_{sh}(S) - H_{sh}(S|R)) = -H(S|R) + H_{sh}(S|R)$$

where we have used the fact that shuffling does not affect the stimulus probabilities and so $H_{sh}(S) = H(S)$. Now

$$\begin{aligned}
\Delta I_{sh} &= -H(S|R) + H_{sh}(S|R) \\
&= -\sum_{r,s} P(r,s)\log\left(P(s|r)\right) + \sum_{r,s} P_{sh}(r,s)\log\left(P_{sh}(s|r)\right) \\
&= -\sum_{r} P(r)\sum_{s} P(s|r)\log\left(P(s|r)\right) + \sum_{r,s} P(s,r)\log\left(P_{sh}(s|r)\right) - \ldots \\
&\qquad\qquad \sum_{r,s} P(s,r)\log\left(P_{sh}(s|r)\right) + \sum_{r,s} P_{sh}(s,r)\log\left(P_{sh}(s|r)\right) \\
&= -\sum_{r,s} P(s,r)\left(\log\left(P(s|r)\right) - \log\left(P_{sh}(s|r)\right)\right) - \sum_{r,s}\left(P(s,r) - P_{sh}(s,r)\right)\log\left(P_{sh}(s|r)\right) \\
&= \sum_{r} P(r)\sum_{s} P(s|r)\log\left(\frac{P(s|r)}{P_{sh}(s|r)}\right) - \sum_{r,s}\left(P(s,r) - P_{sh}(s,r)\right)\log\left(P_{sh}(s|r)\right) \\
&= I_{corr-dep} + I_{corr-ind} = \Delta I + I_{corr-ind}
\end{aligned}$$

Let's break down this derivation. The first line says that the difference between I and I_{sh} is due to difference in the "noise entropy" for predicting the stimulus from the response for the shuffled and true distributions. (When going from response to stimulus, the term "conditional stimulus uncertainty" might be a better term.) In the next two lines, we have bridged the gap between $H(S|R)$ and $H_{sh}(S|R)$ in two separate steps.

First we take the difference between the number of guesses according the shuffled and true decoding strategies, averaged over the true distribution of stimulus-response probabilities. This is the term ΔI. Writing this term as in the second to last line, shows that ΔI is equal to the Kullback-Leibler divergence between the true condition distribution $P(s|r)$ and the shuffled conditional distribution $P_{sh}(s|r)$ averaged over all stimuli. The Kullback-Leibler divergence can be thought of as a measure of the "distance" from probability distribution to another, and can be shown to be positive. Therefore, ΔI is always positive, confirming our intuition that you can't gain any information by using an alternative decoding strategy that is based on throwing away knowledge about correlations.

Second, the term $I_{corr-ind}$ is seen as measuring difference in number of yes/no guesses to determine the stimulus using the shuffled (independent) coding strategy if one averages over the true distribution vs. the shuffled distribution. One way of thinking about this is that this term depends on whether the true distribution leads to more or less cases where the stimulus is ambiguous given a response as compared to the shuffled distribution.

TO BE EXPLAINED BETTER: For a simple case where responses have a metric, Panzeri et al. show that this term is positive when the stimulus and noise correlations have the same sign and negative when they have a different sign. Also need to explain relation to stimulus-dependent and stimulus-independent noise correlations. I don't really understand this in detail now and am not actually sure that this is exactly right.

Chapter 5

Spiking Neurons

5.1 Rate Models and the Leaky Integrator

THE FOLLOWING IS EXTRACTED FROM A MANUSCRIPT THAT I AM WRITING UP FOR PUBLICATION.

Even though the biophysics of spike generation were worked out nearly 50 years ago, surprisingly little is understood about how neurons convert dynamically changing patterns of synaptic input into output spike trains. As a result, most rate-based models take the following form:

$$\tau \dot{u} = -u + h(r_1, r_2, \ldots, r_N) \tag{5.1}$$

$$r = f(u) \tag{5.2}$$

r_i represents the firing rate of the ith presynaptic neuron, and u is some internal "activation variable." h is the function that determines the internal state in response to inputs arriving at rates r_i and f is an input/output function that converts u into an output firing rate r. The dynamics of encoding are determined by the single time constant τ. Based on ad hoc arguments, τ is generally tied to a single biological parameter. Most commonly, τ is assumed to correspond to the membrane time constant (Wilson and Cowan, 1973). More recently, it has been argued that τ corresponds to the time scale of synaptic currents (Knight, 1972; Frolov and Medvedev, 1986; Koch, 1999). A third possibility is that τ may be related to the length of the refractory period (Wilson and Cowan, 1972; Abeles, 1991).

Two main approaches have been taken to understanding the conversion of synaptic input into firing rates. In the first approach, synaptic input is assumed to take the form of a slowly varying input current or conductance. Using this simplifying assumption, quite general procedures have been developed for reducing biophysically realistic models to simpler firing rate models (*e.g.* **??**). An alternative approach has been to assume that the biophysics of spike generation cam be well-approximated by a simple threshold-crossing criterion. Under the additional assumption that individual synaptic events are small relative to threshold, classic tools from stochastic process theory can be used to determine the distribution of interspike intervals (reviewed in **?**). Again, input rates are assumed to be slowly varying so that the arrival statistics of synaptic input are approximately stationary. Recently, this stochastic framework has been extended to cases including dynamically varying inputs **?**Gerstner (2000); Knight (2000). In particular, analytic results have been derived by considering fluctuations about a steady-state input rate (see Knight, 2000 for a more general framework).

These notes take a different approach to understanding rate dynamics in simple integrate-and-fire (IF) neurons. The results stem from the viewpoint that IF dynamics stem from a mixture of two important classes of behavior, roughly corresponding to when spike trains are dominated by very small or very large interspike intervals (ISIs), *i.e.* when the neuron is spiking at very high or very low rates (see below). By expressing firing rate in terms of the joint distribution of the membrane voltage and its derivative, it will be shown that for neurons producing exclusively either short or long ISIs, rate responses can be reasonably well-described by closed form expressions similar in form to equations (1) and (2). This approach is limited by the fact that IF models commonly produce a mixture of long and short ISIs, and hence the expressions derived for either regime give an incomplete picture of IF dynamics. However, the approach yields a clear picture of rate encoding at the extremes of IF behavior, and shows that a number of biological time constants may influence the time scale of neural encoding. The resulting intuitions may help to structure more complete investigations of dynamic rate encoding in both real and model neurons.

5.2 Integrate-and-Fire Model

The spiking model used in these notes is a single compartment IF model in which presynaptic spikes result in an exponentially decaying pulse of injected current. The model is based on the standard passive membrane equation,

$$\tau_{\mathrm{m}}\dot{V} = -V + RI^{\mathrm{syn}} \tag{5.3}$$

where R is the membrane resistance, τ_{m} is the membrane time constant, and voltages are expressed relative to the resting potential. The synaptic current

$$I^{\mathrm{syn}} \;=\; \sum_{t_i^{pre}} I_i \exp\left(-(t - t_i^{pre})/\tau_{\mathrm{syn}}\right) \tag{5.4}$$

t_i^{pre} denotes the arrival time of the ith presynaptic input and I_i is the peak synaptic current (positive for excitation, negative for inhibition). Throughout I will use the short-hand notation that $\exp(-t) = e^{-t}$ if t is positive and $\exp(-t) = 0$ if t is negative. When the voltage V reaches threshold ψ, a spike is emitted and the voltage is reset to $V = V^{\mathrm{reset}}$. $\psi = 20\ mV$ and $V^{\mathrm{reset}} = 10\ mV$ in most simulations. For simplicity, only fast synaptic currents are considered ($\tau_{\mathrm{syn}} = 2.5\ msec$), and both excitatory and inhibitory currents are assumed to have the same time course and magnitude (peak amplitude = 0.25 mV). These simplifications are made for ease of presentation only – the analyses readily generalize to models with heterogeneous synaptic parameters.

Because equation (5.3) is linear in the current I^{syn} (problem 5.2.1), it can be integrated to yield the so-called spike-response formalism **?**:

$$V(t) \;=\; \sum_{t_i^{pre}} PSP_i(t - t_i^{pre}) - \sum_{t_k^{post}} AHP(t - t_k^{post}) \tag{5.5}$$

t_k^{post} is the time of the k postsynaptic spike, and the after-spike hyperpolarization (AHP) is given by

$$AHP(t) \;=\; (\psi - V^{\mathrm{reset}})\exp(-t/\tau_{\mathrm{AHP}})$$

with $\tau_{\mathrm{AHP}} = \tau_{\mathrm{m}}$. The function $PSP_i(t)$ gives the time course of a unitary post-synaptic potential (PSP), and takes the shape of a difference of exponentials:

$$PSP_i(t) = I_i R\left(\exp(-t/\tau_{\mathrm{m}}) - \exp(-t/\tau_{\mathrm{syn}})\right) \tag{5.6}$$

The magnitude of PSP_i is proportional to $I_i R$. The synaptic time constant τ_{syn} determines the rise time of the PSP, and the membrane time constant τ_{m} controls the PSP decay.

Problems

Problem 5.2.1 Show the equation (5.3) is linear as a function of current, *i.e.* a superposition of synaptic currents yields a superposition of membrane voltage.

5.3 Two Regimes of IF Behavior

The two basic regimes of IF behavior are illustrated in figure 5.1, which shows simulation results from a neuron receiving Poisson distributed inputs arriving at a constant rate starting at time $t = 0$. When the distribution of input currents is above threshold (the *suprathreshold regime*), spike times are determined by the time it takes to steadily climb to threshold (fig. 5.1A). Neurons operating in this regime act as neural oscillators, producing regular trains of action potentials. Transient input results in synchronous spiking across trials, but accumulating synaptic noise eventually leads to a diffusion in phase and a flat peri-sistimulus time histogram (PSTH). Analysis of this regime (presented below) demonstrates that rate encoding is largely dominated by the synaptic time constant, although more complex dynamics can result from locking and resonance effects. The results largely confirm those from previous studies (Knight, 1972; Gerstner, 2000; Knight, 2000; **?**).

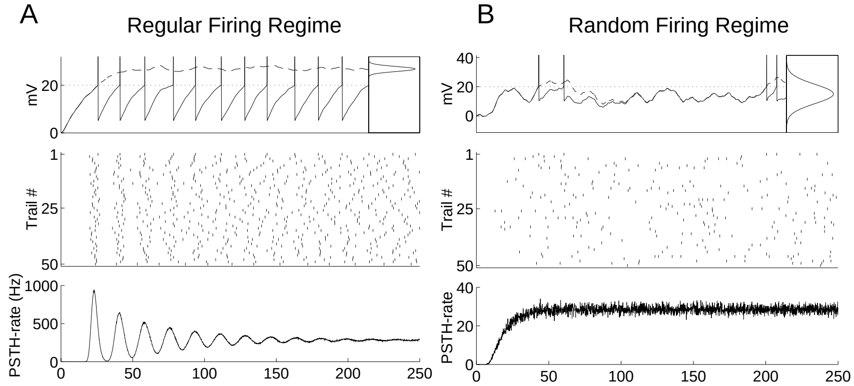


Figure 5.1: Two Regimes of IF Behavior. $\tau_{\mathrm{syn}} = 2.5$ msec; $\tau_{\mathrm{m}} = 15$ msec. Constant levels of input are given to cells starting from rest. Model parameters described in section 5.2. **A.** Subthreshold Regime. Spikes are generated by a monotonic return to spike threshold after the previous spike (*top left*), and the distribution of input current is suprathreshold (*top right, dashed*; plotted in units of voltage - $V = I^{\mathrm{syn}}R - \psi$). Spike rasters from first 50 trials (*middle*) show that spikes are initially highly synchonous, leading to large oscillations in the PSTH (*bottom*) that is eventually damped by accumulating noise. Exc. input: YY kHz; Inh. input: YY kHz; peak PSP size: YY mV; $V^{\mathrm{reset}} = \psi = 20$ mV. **B.** Subthreshold Regime. Spike times are largely determined by random fluctuations in the input current (*top left*), and the bulk of the voltage distribution is below threshold (*top right*). Rasters (*middle*) show asynchonous spiking and a smooth rise in the PSTH (*bottom*). Exc. input: YY kHz; Inh. input: YY kHz; peak PSP size: YY mV; $V^{\mathrm{reset}} = \psi = 10$ mV.

When the bulk of the distribution of membrane voltages remains below threshold (the *subthreshold regime*), spikes result from occasional voltage fluctuations above threshold (fig. 5.1B). Neurons operating in the subthreshold regime produce Poisson-like trains of action potentials, and a transient change in input results in smooth climb to a steady-state rate. Analysis of this regime demonstrates that *both* the membrane *and* synaptic time constants contribute to the time scale of neural encoding. Furthermore, since voltage fluctuations result from randomness in the arrival times of presynaptic spikes (Calvin and Stevens, 1968), the size of the fluctuations is correlated with the presynaptic input rate. This has important dynamic consequences, yielding three additional time constants derived from the membrane and synaptic time scales (see below).

5.4 The PSTH-Rate Revisited

In building a rate model, one first has to choose an appropriate definition of firing rate. These notes focuses on IF neurons receiving stochastic presynaptic input described by a Poisson process. Given the stochastic nature of this input, the membrane voltage will also be stochastic. One can then define spike rate $r(t)$ as the *instantaneous probability that the voltage* $V(t)$ *crosses spike threshold* ψ, where the probability is computed over the distribution of presynaptic spike trains. This rate can be termed the *PSTH-rate* since it can be estimated by calculating the PSTH from many instantiations of the stochastic input stimulus. Experimentally, this histogram can be estimated from many repetitions of a stimulus to a single neuron, or from the activity within a population of neurons having similar response properties. In the rest of these notes we will refer to the PSTH-rate simply as the rate.

Assuming that spikes are caused by the membrane voltage crossing a fixed threshold, the PSTH-rate can be written as follows:

$$r(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \mathcal{P}\left\{ V(t) < \psi \ \& \ \Delta t \dot{V}(t) > \psi - V(t) \right\} \tag{5.7}$$

In other words, $r(t)$ is the instantaneous probability that the voltage is below threshold at time t and the derivative of the voltage is large enough to push the voltage over threshold in the infinitessimal interval Δt. Plotting the voltage on the horizontal axis and the voltage derivative on the vertical, the condition for a spike shows up as the gray region in figure 5.2. Armed only with the definition and some mild boundedness conditions on the derivative of the membrane voltage, it can be shown that

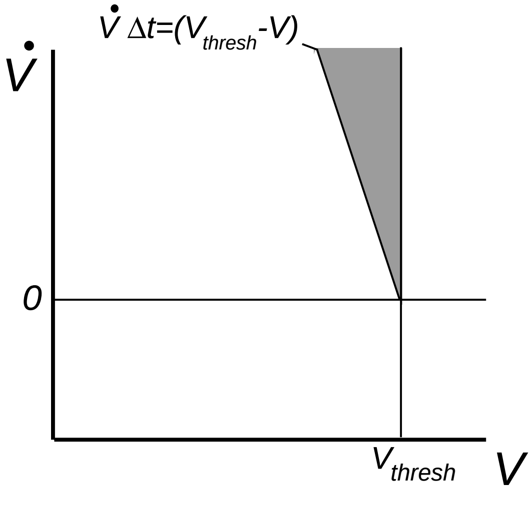


Figure 5.2: Region of $(V, \dot{V})$ space leading to a spike in time Δt.

$$r(t) \quad = \quad \underbrace{\mathrm{P}_{V(t)}(\psi)}_{\text{“voltage term”}} \quad \times \quad \underbrace{\left\langle \left[\dot{V}(t)\right]^{+} \middle| \ V(t) = \psi \right\rangle}_{\text{“current term”}} \tag{5.8}$$

where $\mathrm{P}_{V(t)}(\psi)$ is the probability density function of V evaluated at ψ, and $\left\langle \left[\dot{V}(t)\right]^{+} \middle| \, V(t) = \psi \right\rangle$ is the expected value of the rectified derivative conditioned on $V(t) = \psi$ (see section **??** for proof). $\langle\rangle$ denotes an ensemble average over the distribution of stochastic inputs. The argument is quite general and equation (5.9) can be applied to a wide range of neural models. Intuitively, the equation says that in order to spike (i) the membrane voltage must be near threshold and (ii) given that the voltage starts near threshold, spike rate is proportional to the average rate that the voltage crosses threshold. For the specific case of the passive single compartment neuron (equation 5.3), if $V(t) = \psi$, then the derivative of the voltage $\dot{V} = \psi - IR$. Then, the second term in equation (5.9), $\left\langle \left[\dot{V}(t)\right]^{+} \middle| \, V(t) = \psi \right\rangle$, depends only on the synaptic *current*, whereas the first term, $\mathrm{P}_{V(t)}(\psi)$, depends only on the membrane *voltage*. In other words, *changes in* **both** *the voltage* **and** *the current may contribute to the time course of neural processing, and these two factors should interact multiplicatively.*

5.5 The Suprathreshold Regime

In the suprathreshold regime, the membrane potential increases monotonically, punctuated by spikes and rapid hyperpolarizations. The positivity of the derivative is the key feature of suprathreshold behavior. Given a positive derivative, the current term in the rate equation (5.9) is given by

$$\left\langle \left[\dot{V}(t)\right]^{+} \middle| \, V(t) = \psi \right\rangle = \left(\langle I(t)\rangle R - \psi\right)/\tau_{\mathrm{m}} \tag{5.9}$$

Therefore, the current term depends on the *mean* input current, and hence reacts to changes in presynaptic spike rate on the time scale of the synaptic time constant τ_{syn}. Positivity of the derivative also confines the voltage to lie between the reset voltage V^{reset} and the threshold ψ.

To analyze the voltage term, $\mathrm{P}_{V(t)}(\psi)$, we will take advantage of the oscillatory nature of spiking in the suprathreshold regime. First, consider the case of constant injected current $\bar{I}$. The neuron oscillates at a firing frequency $f(\bar{I})$. The plot of f as a function of I is the so-called "f-I curve." For an IF neuron,

$$f(I) = \left(-\tau_{\mathrm{m}} \log \frac{IR - \psi}{IR - V^{\mathrm{reset}}}\right)^{-1} \tag{5.10}$$

(problem 5.5.1). Suppose we examine the neuron at random phases of the oscillation, and let $\mathrm{P}_{Vss}(V, \bar{I})$ denote the "steady-state" probability that the membrane voltage has a particular value V. Intuitively, $\mathrm{P}_{Vss}(V, \bar{I})$ is inversely proportional to how fast the voltage is rising past V. For an IF neuron, $\mathrm{P}_{ss}(V, \bar{I})$ can be explicitly determined:

$$\mathrm{P}_{Vss}(V, \bar{I}) = \frac{1}{\bar{I}R - V} \left(\log\left(\frac{\bar{I}R - V^{\mathrm{reset}}}{\bar{I}R - \psi}\right)\right)^{-1} \tag{5.11}$$

(problem 5.5.1). Note that since the voltage change slows as the cell decays back to the equilibrium voltage IR, $\mathrm{P}_{Vss}(V, \bar{I})$ is skewed toward threshold (figure 5.3A). The degree of skew depends on how close IR is to threshold, and hence indirectly depends on spike rate. At high rates, the return to threshold is nearly linear, and P_{Vss} is nearly flat (figure 5.3A, *left*). At low rates, there is significant slowing of the voltage derivative as the voltage approaches threshold, and the distribution P_{Vss} is skewed toward (figure 5.3A, *right*).

To calculate the true voltage distribution, we multiply $\mathrm{P}_{Vss}(V,\bar{I})$ multiplied by the distribution of phases of the ongoing oscillation at time t, $\rho(s,t)$, *i.e.*

$$\mathrm{P}_{V(t)}(V) = \rho(s,t)\mathrm{P}_{Vss}(V,\bar{I}) \tag{5.12}$$

The phase variable s $(0 \leq s < 1)$ is defined as the time since the last spike, expressed as a fraction of the cycle time $T = 1/f(I)$, *i.e.* $s = (t - t_{spike})f(\bar{I})$. $\rho(s,t)$ describes the probability of being at phase s at time t. Therefore, for constant current I, $\rho(s,t)$ describes the probability that a spike will occur at time $t + (1-s)f(\bar{I})$.

Now we consider the case of a time varying, stochastic input current $I(t)$. This input will lead to a distribution $\mathrm{P}_{V(t)}(V)$ of voltages at any time t. Treating the neuron at any time t like an oscillator driven by the average current $\langle I(t)\rangle$, we use equation (5.12) to describe the voltage distribution as the product of a steady state distribution and a phase distribution, *i.e.* we *define* the phase distribution as $\rho(s,t) = \mathrm{P}_{V(t)}(V)/\mathrm{P}_{Vss}(V,\langle I(t)\rangle)$. Combining equation (5.12) with the expression for the current term in the suprathreshold regime (equation 5.9), we find that

$$\begin{aligned} r(t) &= \rho(s,t)\mathrm{P}_{Vss}(\psi,\bar{I})(\langle I(t)\rangle R - \psi)/\tau_{\mathrm{m}} \\ &= \frac{\rho(s,t)}{\langle I(t)\rangle R - \psi}\left(\log\left(\frac{\langle I(t)\rangle R - V^{\mathrm{reset}}}{\langle I(t)\rangle R - \psi}\right)\right)^{-1}(\langle I(t)\rangle R - \psi)/\tau_{\mathrm{m}} \\ &= \rho(s,t)f(\langle I(t)\rangle) \end{aligned} \tag{5.13}$$

This equation implies that we can view rate encoding in the suprathreshold regime as arising from two effects: the firing rate driven the mean current at time t and the modulation in rate due to phase synchrony effects. Because the neuron is acting like an oscillator, phase synchrony will result in an oscillation of the PSTH-rate at a frequency equal to the underlying firing frequency $f(\langle I(t)\rangle)$. *Simply assuming no phase synchrony in the suprathreshold regime* ($\rho(s,t) = 1$) *leads to a very simple rate model: (i) simply compute the mean synaptic current, and (ii) set the firing rate according to that current* (Koch, 1999). Phase synchrony in response to dynamic inputs can result in complex resonances (Knight, 1972; Gerstner, 2000). We refer the reader to Knight (2000) for an elegant approach to the analyzing these effects in a number of interesting cases.

Examples of rate behavior in the suprathreshold regime are shown in figure 5.3. Input rates were chosen so that fing rate undergoes a rapid change from 67 Hz to YY Hz, followed by a period of rapidly changing "noisy" input rate. The lower trace shows the difference between the simulated PSTH and the PSTH-rate predicted by the simple rate model. The step change induces phase synchrony which results in an oscillation of the PSTH-rate at the firing frequency of the neuron. This dies out with accumulating noise. While the simple model does a reasonable model of predicting the dynamic response to rapidly changing input rates, phase synchrony results in prediction errors that are of the same order of magnitude as the oscillations induced by a step transient.

While we will not go into the analysis of phase synchrony, equation (**??**) will be used to emphasize two important intuitions. First, while the overall rate is controlled by the mean current, variance in the input current reduces the contribution of the phase term. Higher variance means more rapid diffusion in phase and hence more rapidly damped oscillation. The second, less obvious point to be drawn from equation (**??**) is that the degree of phase synchrony depends on overall spike rate. Phase synchrony results when changing input rates change the voltage distribution $\rho(V,I)$. More specifically, assume a step change in mean input current from I_{pre} to a new level I_{post}. Assuming no phase sychrony before the change ($\rho_{pre}(s,t) = 1$), the distribution of voltages is given by $\mathrm{P}_{V(t)}(V) = \mathrm{P}_{Vss}(V,I_{pre})$. Immediately after the change, the actual voltage distribution

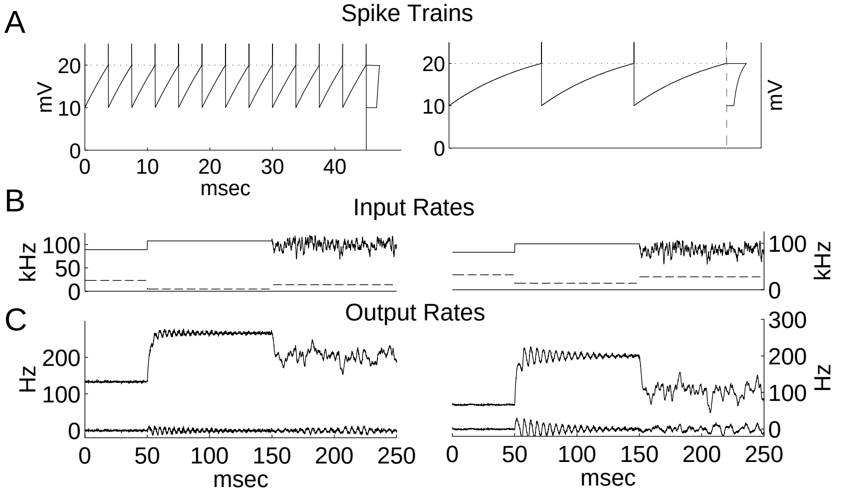


Figure 5.3: Suprathreshold Behavior at High (right) and Very High (left) Spike Rates. **A.** Spike trains with constant input current. Plot at right shows the probability distribution of membrane voltage. At very high rates, the distribution is nearly flat. At lower rates, the neuron spends more time near threshold. **B.** Time varying inputs. At 50 msec excitatory input rates (solid) and inhibitory input rates (dashed) are stepped in opposite directions to give a step increase in firing rate (Left: excitation - YY kHz/ZZ kHz; inhibition - YY kHz/ZZ kHz. Right: excitation - YY kHz/ZZ kHz; inhibition - YY kHz/ZZ kHz.) Total input rate (exc.+inh.) is constant over first 150 msec. At 150 msec, mean inputs are stepped to levels half-way between the previous conditions, and excitatory input rates are modulated by unstructured noise (Gaussian noise with std = YY kHz, sampled at 0.1 msec intervals and then smoothed by a 1 msec wide square sliding average.) **C.** Top: Output spike rates. Bottom: Residual after subtracting prediction of theoretical model obtained from the current value of the mean input current. Input paramters were chosen so that mean output spike rates differed by a fixed amount (67 Hz) during periods of contant input. The PSTH-rate dynamics is dominated by the synaptic current (and time constant), but transients give rise to phase locking. Phase locking is more pronounced at lower spike rates (standard deviation of residual: NN Hz - left; MM Hz - right).

$\mathrm{P}_{V(t)}(V)$ hasn't changed. Equation (**??**) then implies

$$\mathrm{P}_{Vss}(V, I_{pre}) \;=\; \mathrm{P}_{V(t)}(V) \;=\; \rho_{post}(s,t)\mathrm{P}_{Vss}(V, I_{post}) \tag{5.14}$$

Therefore

$$\rho_{post}(s,t) = \mathrm{P}_{Vss}(V, I_{pre})/\mathrm{P}_{Vss}(V, I_{post}) \tag{5.15}$$

Recall that at very high rates, the voltage distribution is nearly flat. Therefore $\rho(V, I_{pre}) \approx \rho(V, I_{post})$ and the phase distribution $\rho_{post}(s)$ remains flat, even after a transient. This effect is demonstrated in figure 5.3B and C. The inputs in the example on the left and right have the same overall firing rate. However, the example on the the left has a greater proportion of exicatory inputs, leading to increased firing rates. As expected from equation 5.15 phase synchrony is reduced at higher rates.

Problems

Problem 5.5.1 Derive equations (5.9), (5.10), (5.11), and (5.13).

5.6 The Subthreshold Regime

The subthreshold regime is defined by the condition that spikes are caused by occaisional synaptic fluctuations. Recovery from the previous spike makes a negligible contribution to spiking. Rate encoding in this regime can be analyzed by simply ignoring the AHP term in equation (5.5) and assuming that the voltage is governed exclusively by the synaptic term, *i.e.*

$$V(t) \quad = \quad \sum_{t_i^{pre}} PSP_i(t - t_i^{pre}) \tag{5.16}$$

This leads to a simple "threshold-crossing" model where spikes are registered when the voltage crosses threshold from below, but *the voltage is not reset and is allowed to drift above threshold after each spike* (fig. 5.4A). This assumption leads to a peculiar form of refractoriness, since the model is unable to spike until the voltage descends below threshold. However, because the distribution of inputs is assumed to be largely subthreshold, this non-realistic refractory behavior has only a minor influence on spiking.

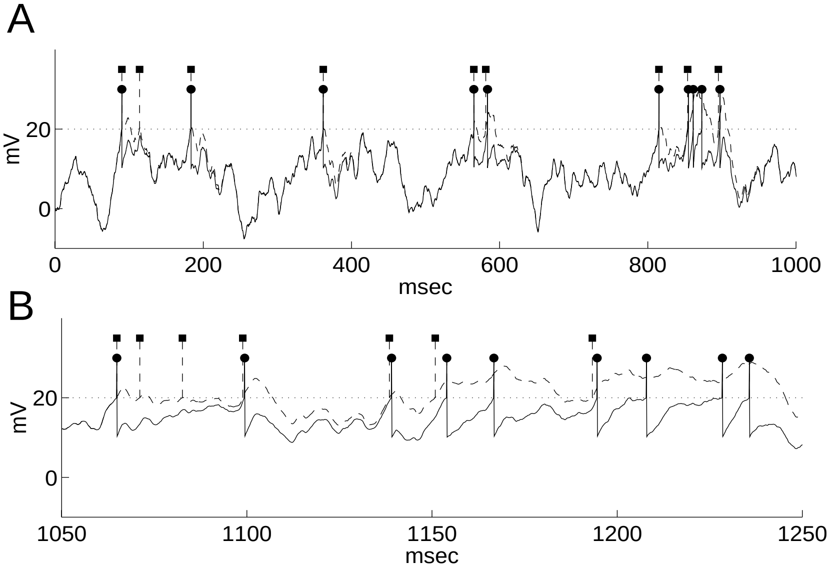


Figure 5.4: Threshold-crossing Model. Voltage traces from IF model (*solid*) and threshold-crossing model (*dahsed*) in the subthreshold regime (excitation - YY kHz/ZZ kHz; inhibition - YY kHz/ZZ kHz). Spikes from IF model are marked by circles, those from threshold-crossing model with squares. **A.** First 1 sec of simulation. Spiking responses of two models are similar though not identical. **B.** Blow up of 200 msec selected to highlight the difference between the two models. From 1050-1100 msec, the synaptic term is hovering near threshold, yielding a series of spikes in the threshold-crossing model. The AHP from the first spike prevents spiking in the IF model. From 1200-1250 msec, the synaptic term remains largely above threshold, yielding a series of spikes in the IF model but only a single spike in the threshold-crossing model.

Figure 5.4B shows a stretch of response that has been selected to highlight the difference between IF and threshold-crossing model behavior. Bursts of spikes are produced in the threshold-crossing model when the synaptic input is hovering near threshold (1050-1100 msec), whereas spike reset prevents this burst in the IF model. Conversely, bursts of spikes occur in the IF model when the synaptically driven voltage remains above threshold. The threshold-crossing model produces a spike only during the initial crossing of threshold. Even though the two models display different burst behavior, most spikes are produced after relatively long interspike intervals. Thus, the majority of the spikes produced by the two models are closely aligned (fig. 5.4A).

The advantage of considering the threshold-crossing model is that it can be treated analytically by adopting one additional simplification. Thus far, it has been assumed that the arrival of presynaptic spikes is governed by a Poisson process. Assuming that the unitary PSPs are small, this process is well-approximated by a Gaussian process having the same mean and variance. This is the standard "diffusion approximation" for stochastic differential equations **??**. Under this assumption, the joint distribution of the voltage and the voltage derivative is a two-dimensional Gaussian. For example, the mean of the voltage distribution at time t, $\mu_{\mathrm{V}}(t)$, is obtained by integrating the contribution from all previous intervals of width dt to the current value of the voltage, *i.e.*

$$\mu_{\mathrm{V}}(t) \;=\; \sum_c \int_{-\infty}^{t} ds\; r_c(s) PSP_n(t-s) \tag{5.17}$$

where $r_c(t)$ is the rate of the cth class of presynaptic input with unitary PSP shape PSP_n. All simulations presented contain just two classes of inputs, one excitatory and one inhibitory, and the PSP shapes differ only in sign. Similarly,

$$\begin{aligned}
\nu_{\mathrm{V}}(t) &= \sum_c -\mu_{\mathrm{V}}^2(t) + \int_{-\infty}^{t} ds\; r_c(s) PSP_c^2(t-s) && (5.18)\\
\mu_{\dot{\mathrm{V}}}(t) &= \sum_c \int_{-\infty}^{t} ds\; r_c(s) P\dot{S}P_c(t-s) && (5.19)\\
\nu_{\dot{\mathrm{V}}}(t) &= \sum_c -\mu_{\dot{\mathrm{V}}}^2(t) + \int_{-\infty}^{t} ds\; r_c(s) P\dot{S}P_c^2(t-s) && (5.20)\\
\nu_{\mathrm{V},\dot{\mathrm{V}}}(t) &= \sum_c -\mu_{\mathrm{V}}(t)\mu_{\dot{\mathrm{V}}}(t) + \int_{-\infty}^{t} ds\; r_c(s) PSP_c(t-s) P\dot{S}P_c(t-s) && (5.21)
\end{aligned}$$

where $\mu_{\dot{\mathrm{V}}}$ is the mean of the derivative of the voltage, ν_{V} and $\nu_{\dot{\mathrm{V}}}$ are the variances of the voltage and the derivative of the voltage respectively, and $\nu_{\mathrm{V},\dot{\mathrm{V}}}$ is the covariance between the voltage and the derivative of the voltage. $P\dot{S}P$ is the time derivative of the PSP. It follows that the derivative of the voltage, conditioned on $V(t) = \psi$ is a Gaussian distribution with mean $\mu_{\psi,\dot{\mathrm{V}}} = \mu_{\dot{\mathrm{V}}} + (\psi - \mu_{\mathrm{V}})\nu_{\mathrm{V},\dot{\mathrm{V}}}/\nu_{\mathrm{V}}$ and variance $\nu_{\psi,\dot{\mathrm{V}}} = \nu_{\dot{\mathrm{V}}} - \nu_{\mathrm{V},\dot{\mathrm{V}}}^2/\nu_{\mathrm{V}}$. Plugging these parameters into equation (5.9) yields the following expression for the PSTH-rate in the subthreshold regime:

$$r(t) \;=\; \frac{1}{\sqrt{2\pi\nu_{\mathrm{V}}}} \exp\left(\frac{-(\mu_{\mathrm{V}}-\psi)^2}{\nu_{\mathrm{V}}}\right) \times \frac{1}{\sqrt{2\pi\nu_{\psi,\dot{\mathrm{V}}}}} \int_0^{\infty} d\dot{V}\; \dot{V} \exp\left(\frac{-(\dot{V}-\mu_{\psi,\dot{\mathrm{V}}})^2}{\nu_{\psi,\dot{\mathrm{V}}}}\right) \tag{5.22}$$

Figure 5.5 shows that match between rates derived from equation (5.22) (*thick line*) and the PSTH derived from multiple simulations of IF model (*thin lines*). Reset voltage was 10 mV. CHECK The analytic model provides a reasonably accurate prediction of IF dynamics, matching the rise in response to a step transient (figure 5.5B) as well as the response to rapidly varying input rates on submillisecond time scale (figure 5.5C).

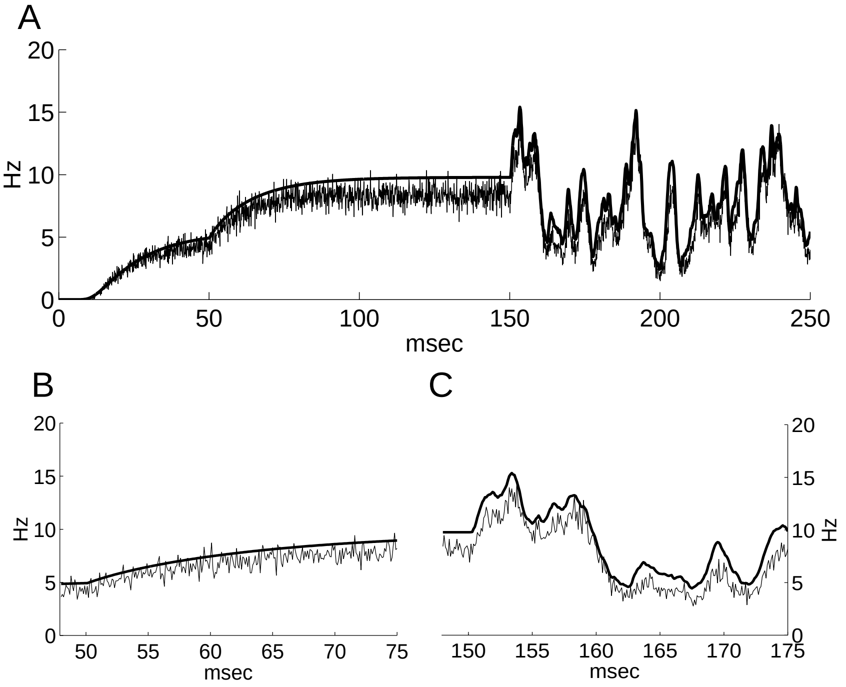


Figure 5.5: Match of Analytic Threshold-crossing Model to IF Behavior.

While it appears to have a relatively minor influence on rate *dynamics*, varying the size of the after-spike hyperpolarization in the IF model does alter the overall firing rate. Small AHPs lead to higher spike rates, and large AHPs lead to lower spike rates. Since the threshold-crossing model completely ignores the AHP, this dependence is not captured by the rate equation (5.22). The dependence of spike rate on the AHP implies that the AHP term *does* affect spiking to some degree, even at rates of 10 Hz and lower. This is somewhat surprising, given that a 10 Hz spike rate implies that the mean interspike interval is 100 *msec*, over six times as long as the 15 *msec* decay time of the AHP. The effect of the AHP can be understood, however, by remembering that in the subthreshold regime model behavior resembles that of a Poisson process. Consider the exponential of interspike interval distribution produced by a Poisson process firing at 10 Hz. Even though the mean interval is 100 *msec*, the median interval is 69.3 *msec*, and *nearly 14% of intervals fall within the decay constant of 15 msec.* Thus, it should expected that the AHP term will have *some* effect on the number of short interspike intervals, well into what could be considered the subthreshold regime. "Pure" subthreshold behavior is expected only at the very lowest spike rates.

5.6.1 Multiple Time Scales

While the ability to write down a closed form expression for the spike rate in the subthreshold regime is satisfying, the real value of equations (5.17)-(5.22) is the insight gained regarding rate encoding. For example, in the subthreshold regime it is evident that both the mean and the variance of the synaptic input play a role in determining the output rate. This is true both in the current term, where the variance of the voltage derivative enters via the error function, and the voltage term where the probability that $V(t) = \psi$ depends on both the mean and the variance of the voltage distribution. More importantly, consideration of the variance of the input as well as the mean adds a number of additional time scales to the problem.

To understand the contribution of these multiple time scales, we first consider a very simple example of a neuron receiving input from a single population of presynaptic neurons. Inputs from this population are assumed to arrive at a mean rate $r(t)$. Each of these neurons is assumed to give rise to a synaptic current that is described by an instantaneous rise and exponential decay with decay time τ_{syn}. According to equation (5.6) the PSP can be written as a difference of exponentials:

$$PSP(t) \quad = \quad I_i R\left(\exp(-t/\tau_{\mathrm{m}}) - \exp(-t/\tau_{\mathrm{syn}})\right)$$

But then to calculate the mean potential, equation (5.17) becomes

$$\begin{aligned}
\mu_{\mathrm{V}}(t) &= \int_{-\infty}^{t} ds\ r(s) I_i R\left(\exp(-(t-s)/\tau_{\mathrm{m}}) - \exp(-(t-s)/\tau_{\mathrm{syn}})\right) && (5.23)\\
&= I_i R\left(\int_{-\infty}^{t} ds\ r(s)\exp(-(t-s)/\tau_{\mathrm{m}}) - \int_{-\infty}^{t} ds\ r(s)\exp(-(t-s)/\tau_{\mathrm{syn}})\right) && (5.24)\\
&= I_i R\left(\tau_{\mathrm{m}} r_{\tau_{\mathrm{m}}}(t) - \tau_{\mathrm{syn}} r_{\tau_{\mathrm{syn}}}(t)\right) && (5.25)
\end{aligned}$$

where $r_\tau(t)$ is the rate of presynaptic input smoothed with time constant τ, *i.e.*

$$r_\tau(t) = \frac{1}{\tau}\int_{-\infty}^{t} ds\ r(s)\exp(-(t-s)/\tau) \qquad (5.26)$$

From equation (5.25) we see that both the synaptic and membrane time constants enter into the calculation of the mean potential. However, if we consider synaptic currents that are much faster than the membrane time constant ($\tau_{\mathrm{syn}} << \tau_{\mathrm{m}}$), then $\tau_{\mathrm{m}} r_{\tau_{\mathrm{m}}}(t) >> \tau_{\mathrm{syn}} r_{\tau_{\mathrm{syn}}}(t)$ and the membrane time constant dominates the calculation of the mean voltage. This case models the most common intuition regarding synaptic integration: the membrane time constant is the limiting factor that determines the time window over which synaptic inputs are integrated. Note that many experiments suggest that this may not be the only, or even dominant mode of synaptic integration. The membrane time constant is reduced when a neuron receives a large barrage of synaptic input, and there is ample evidence that "slow" synaptic currents (where τ_{syn} is approximately the equal to or longer than τ_{m}) contribute large synaptic inputs, at least in the cortex.

To calculate the mean of the voltage derivative, we must take the time derivative of the PSP and plug it in to equation (5.19):

$$\begin{aligned}
P\dot{S}P(t) &= I_i R\left(\exp(-t/\tau_{\mathrm{m}})/\tau_{\mathrm{m}} - \exp(-t/\tau_{\mathrm{syn}})/\tau_{\mathrm{syn}}\right) && (5.27)\\
\mu_{\mathrm{V}}(t) &= \int_{-\infty}^{t} ds\ r(s) I_i R\left(\exp(-(t-s)/\tau_{\mathrm{m}})/\tau_{\mathrm{m}} - \exp(-(t-s)/\tau_{\mathrm{syn}})/\tau_{\mathrm{syn}}\right) && (5.28)\\
&= I_i R\left(\int_{-\infty}^{t} ds\ r(s)\exp(-(t-s)/\tau_{\mathrm{m}})/\tau_{\mathrm{m}} - \int_{-\infty}^{t} ds\ r(s)\exp(-(t-s)/\tau_{\mathrm{syn}})/\tau_{\mathrm{syn}}\right) && (5.29)\\
&= I_i R\left(r_{\tau_{\mathrm{m}}}(t) - r_{\tau_{\mathrm{syn}}}(t)\right) && (5.30)
\end{aligned}$$

In calculating the mean of the voltage derivative, the synaptic and membrane time constants play an equal role.

Now let's calculate the variance of the voltage. First we define a "mixed" time constant $\tau_{\mathrm{mix}} = \tau_{\mathrm{m}}\tau_{\mathrm{syn}}/(\tau_{\mathrm{m}} + \tau_{\mathrm{syn}})$ that will crop up in the calculations below. According to equation (5.18) we have

$$\nu_{\mathrm{V}}(t) \quad = \quad -\mu_{\mathrm{V}}^2(t) + \int_{-\infty}^{t} ds\ r(s)(I_i R)^2\left(\exp(-(t-s)/\tau_{\mathrm{m}}) - \exp(-(t-s)/\tau_{\mathrm{syn}})\right)^2 \qquad (5.31)$$

$$
\begin{aligned}
&= -\mu_{\mathrm{V}}^2(t) + (I_i R)^2 \left(\int_{-\infty}^{t} ds\ r(s) \exp(-2(t-s)/\tau_{\mathrm{m}}) - \right. \\
&\qquad \left. \int_{-\infty}^{t} ds\ 2r(s) \exp(-(t-s)/\tau_{\mathrm{mix}}) + \int_{-\infty}^{t} ds\ r(s) \exp(-2(t-s)/\tau_{\mathrm{syn}}) \right) \qquad (5.32) \\
&= -(I_i R)^2 \left(\tau_{\mathrm{m}}^2 r_{\tau_{\mathrm{m}}}^2(t) - 2\tau_{\mathrm{m}}\tau_{\mathrm{syn}} r_{\tau_{\mathrm{m}}}(t) r_{\tau_{\mathrm{syn}}}(t) + \tau_{\mathrm{syn}}^2 r_{\tau_{\mathrm{syn}}}^2(t) + \right. \\
&\qquad \left. \frac{\tau_{\mathrm{m}}}{2} r_{\tau_{\mathrm{m}}/2}(t) - 2\tau_{\mathrm{mix}} r_{\tau_{\mathrm{mix}}}(t) + \frac{\tau_{\mathrm{syn}}}{2} r_{\tau_{\mathrm{syn}}/2}(t) \right) \qquad (5.33) \\
&\qquad (5.34)
\end{aligned}
$$

From this equation we see that there are actually five distinct time constants that affect $\nu_{\mathrm{V}}(t)$: $\tau_{\mathrm{m}}, \tau_{\mathrm{syn}}, \tau_{\mathrm{m}}/2, \tau_{\mathrm{syn}}/2, \tau_{\mathrm{mix}}$. The latter three arise because of the squaring operation used to calculate the variance. Similar calculations can be performed for $\nu_{\dot{\mathrm{V}}}(t)$ and $\nu_{\mathrm{V},\dot{\mathrm{V}}}(t)$ (problem **??**).

Problems

Problem 5.6.1 Calculate $\nu_{\dot{\mathrm{V}}}(t)$ and $\nu_{\mathrm{V},\dot{\mathrm{V}}}(t)$ for the case of a single population of presynaptic neurons having a synaptic time constant τ_{syn}.

5.7 The Intermediate Regime

These notes focus on an analysis of two extremes of IF behavior. Except at the very high or very low rates, neurons are expected to display a mixture of supra- and subthreshold behaviors (the intermediate regime). For example, short interspike intervals are expected to be governed by suprathreshold behavior since they arise when synaptic input wanders above threshold. Conversely, long intervals are expected to be governed by subthreshold behavior. To truly understand rate encoding in IF neurons, it will be necessary to gain a better understanding of the statistics governing the switching between these two regimes.

A number of mechanisms are expected to affect the relative balance between supra- and subthreshold behaviors. For example, experimental evidence suggesting that spike trains produced by neocortical neurons have highly variable ISI statistics Softky and Koch (1993); **?** has lead a number of authors to examine the mechanisms contributing to subthreshold behavior. Because a variety of biological mechanisms can influence the size of synaptic fluctuations *relative* to the magnitude of the AHP term, a number of factors can contribute to the relative contribution of sub- and suprathreshold behavior. One mechanism that has received much attention is the balance between excitatory and inhibitory inputs **??**. A neuron receiving a large number of inhibitory inputs will require a large number of excitatory inputs in order to spike. Since the variance in the synaptic current increases with the *total* number of inputs, inhibitory balance will lead to large fluctuations in the synaptic term. Alternatively, network dynamics may increase fluctuations in the synaptic by causing fluctuations in the spike rates of presynaptic neurons **??**. Another factor influencing the relative magnitude of the synaptic fluctuations and the AHP term is the strength of the AHP. Thus, small AHPs contribute to subthreshold behavior **?**Troyer and Miller (1997). While all of these factors may play a role, the strongest influence on whether neurons operate in the high or suprathreshold regimes is likely to be spike rate. At high rates, many spikes contribute to the AHP term and this term is large, leading to suprathreshold behavior. At low rates, the AHP term decays away and spikes result from fluctuations in the synaptic current.

5.8 Proof of the Main Result

Consider the joint probability density function $\mathcal{P}(V,\dot{V})$. Let $\mathcal{P}_V(V)$ and $\mathcal{P}_{\dot{V}}(\dot{V})$ be the probability density functions of V and $\dot{V}$, respectively, and let $\mathcal{P}_{V|\dot{V}_0}(V)$ be the probability density of V conditioned on $\dot{V}=\dot{V}_0$. Suppose that (i) the partial derivative $|d\mathcal{P}_{V|\dot{V}_0}(V)/dV| \leq K$ for all $(V,\dot{V})$ in the half plane defined by $V \leq \psi$, (ii) $\lim_{x\to\infty} x \int_x^\infty d\dot{V}\ \mathcal{P}_{\dot{V}}(\dot{V}) = 0$, and (iii) $\int_{-\infty}^{\infty} d\dot{V}\ \dot{V}\ \mathcal{P}_{\dot{V}}(\dot{V}) = 0$ Then, the instantaneous probability of reaching spike threshold,

$$\begin{aligned} r(t) &= \lim_{\Delta t\to 0} \frac{1}{\Delta t}\ \mathcal{P}\left\{V(t) < \psi \ \&\ \Delta t\dot{V}(t) > \psi - V(t)\right\} \\ &= \mathcal{P}_V(\psi)\ \left\langle \left[\dot{V}\right]^+ \middle|\ V=\psi\right\rangle \end{aligned}$$

Proof:

r can be rewritten as

$$r(t) \ = \ \lim_{\Delta t\to 0} \frac{1}{\Delta t}\int_0^\infty d\dot{V}\ \int_{\psi-\Delta t\dot{V}}^{\psi} dV\ \mathcal{P}(V,\dot{V})$$

(see fig. **??**). Fix $\epsilon > 0$ and break the integral into two terms, depending on whether $\dot{V}$ is large enough to force the voltage $V=\psi-\epsilon$ past threshold, *i.e.*

$$r(t) \ = \ \lim_{\Delta t\to 0} \frac{1}{\Delta t}\left(\int_0^{\frac{\psi-\epsilon}{\Delta t}} d\dot{V}\ \int_{\psi-\Delta t\dot{V}}^{\psi} dV\ \mathcal{P}(V,\dot{V}) + \int_{\frac{\psi-\epsilon}{\Delta t}}^{\infty} d\dot{V}\ \int_{\psi-\Delta t\dot{V}}^{\psi} dV\ \mathcal{P}(V,\dot{V})\right)$$

By setting $x = (\psi-\epsilon)/\Delta t$ and noting that $\int_{\psi-\Delta t\dot{V}}^{\psi} dV\ \mathcal{P}(V,\dot{V}) \leq \int_{-\infty}^{\infty} dV\ \mathcal{P}(V,\dot{V}) = \mathcal{P}_{\dot{V}}(\dot{V})$, condition (ii) implies that the second term goes to zero as $\Delta t \to \infty$. To get a handle on the first term, note that assumption (i), $|d\mathcal{P}_{V|\dot{V}}(V)/dV| \leq K$, is equivalent to the condition $|d\mathcal{P}(V,\dot{V})/dV| \leq K\ \mathcal{P}_{\dot{V}}(\dot{V})$, and $\mathcal{P}(\psi,\dot{V}) = \mathcal{P}_V(\psi)\mathcal{P}_{\dot{V}|\psi}(\dot{V})$, where $\mathcal{P}_{\dot{V}|V_0}(\dot{V})$ denotes the probability density of $\dot{V}$ conditioned on $V=V_0$. Consider voltages where $\psi-\epsilon \leq V \leq \psi$. Then assumption (i) implies that

$$\begin{aligned} \mathcal{P}(V,\dot{V}) &\leq \mathcal{P}(\psi,\dot{V}) + \epsilon K\ \mathcal{P}_{\dot{V}}(\dot{V}) \\ &= \mathcal{P}_V(\psi)\mathcal{P}_{\dot{V}|\psi}(\dot{V}) + \epsilon K\ \mathcal{P}_{\dot{V}}(\dot{V}) \end{aligned}$$

Therefore,

$$\begin{aligned} r(t) &\leq \lim_{\Delta t\to 0} \frac{1}{\Delta t}\int_0^{\frac{\psi-\epsilon}{\Delta t}} d\dot{V}\ \int_{\psi-\Delta t\dot{V}}^{\psi} dV\ \left(\mathcal{P}_V(\psi)\mathcal{P}_{\dot{V}|\psi}(\dot{V}) + \epsilon K\ \mathcal{P}_{\dot{V}}(\dot{V})\right) \\ &= \lim_{\Delta t\to 0} \frac{1}{\Delta t}\int_0^{\frac{\psi-\epsilon}{\Delta t}} d\dot{V}\ \Delta t\ \dot{V}\ \left(\mathcal{P}_V(\psi)\mathcal{P}_{\dot{V}|\psi}(\dot{V}) + \epsilon K\ \mathcal{P}_{\dot{V}}(\dot{V})\right) \\ &= \int_0^\infty d\dot{V}\ \dot{V}\ \left(\mathcal{P}_V(\psi)\mathcal{P}_{\dot{V}|\psi}(\dot{V}) + \epsilon K\ \mathcal{P}_{\dot{V}}(\dot{V})\right) \end{aligned}$$

Since ϵ can be chosen to be arbitrarily small, assumption (iii) implies that

$$r(t) \ \leq \ \mathcal{P}_V(\psi)\int_0^\infty d\dot{V}\ \mathcal{P}_{\dot{V}|\psi}(\dot{V}) \ = \ \mathcal{P}_V(\psi)\ \left\langle \left[\dot{V}\right]^+ \middle|\ V=\psi\right\rangle$$

By reversing the inequalities and substituting $-\epsilon K$ for ϵK, a similar argument shows that $r(t) \geq \mathcal{P}_V(\psi)\ \left\langle \left[\dot{V}\right]^+ \middle|\ V=\psi\right\rangle$, completing the proof.

Chapter 6

Linear and Linear-Nonlinear Neurons

6.1 Intro: Why Study Linear Systems?

These notes will introduce the rudiments of linear systems theory as applied to computations within networks of neurons. But the brain is highly nonlinear. Won't focusing on linear systems give a distorted and perhaps overly simplistic view of neural processing? While this is a danger, there are many reasons to focus on linear systems. The first is entirely practical – the only systems where general techniques provide solutions to wide variety of problems are linear. In studying the linear approximations to brain function we have a host of mathematical tools at our disposal. A second reason is pedagogical – in learning the basic concepts of linear algebra, students will be able to practice the process of putting biological problems into a more abstract framework as well as the process of contemplating the biological implications of insights gained from a more abstract point of view. Finally, and most importantly, a number of our basic notions about how the brain works can be characterized as nearly linear or as "linear-nonlinear." As a result, linear models can go a long way toward clarifying these basic notions. Moreover, one must first understand the linear explanations of neural phenomena before one can grasp the key issues underlying experimental attempts to quantify just how nonlinear the brain is.

What does it mean for a system to be linear? The most basic definition of linearity is that the whole is exactly equal to the sum of its parts. More technically, a system is said to be linear if it has the property of **superposition**. For example, suppose we record from a neuron in the visual cortex when presenting stimuli on a computer screen. The neuron's response function is said to be linear (as a function of luminance) if the response to the combination (or superposition) of stimuli $\mathbf{s}_1$ and $\mathbf{s}_2$ is equal to its response to $\mathbf{s}_1$ plus its response to $\mathbf{s}_2$ (figure 6.1), *i.e.*

$$r(\mathbf{s}_1 + \mathbf{s}_2) = r(\mathbf{s}_1) + r(\mathbf{s}_2) \tag{6.1}$$

Superposition also implies that if we change the strength of a stimulus by multiplying the brightness at each pixel by a scale factor c, we get a corresponding change in the strength of the response:

$$r(c\mathbf{s}_1) = c \times r(\mathbf{s}_1) \tag{6.2}$$

Biological Aside. Note that superposition is a formal version of the common expectation that a mixture of inputs gives a mixture of outputs, and that increasing the magnitude of the cause increases the magnitude of the effect. Any time that these expectations are found – and one runs across them in many neuroscience papers – they imply (nearly) linear thinking.

Pushing our example a bit further reveals why linear systems are so easy to analyze. Suppose every stimulus that we presented could be written as a **linear combination** of a finite number stimuli, $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\}$. That is

$$\mathbf{s} = c_1\mathbf{s}_1 + c_2\mathbf{s}_2 + \ldots + c_N\mathbf{s}_N \tag{6.3}$$

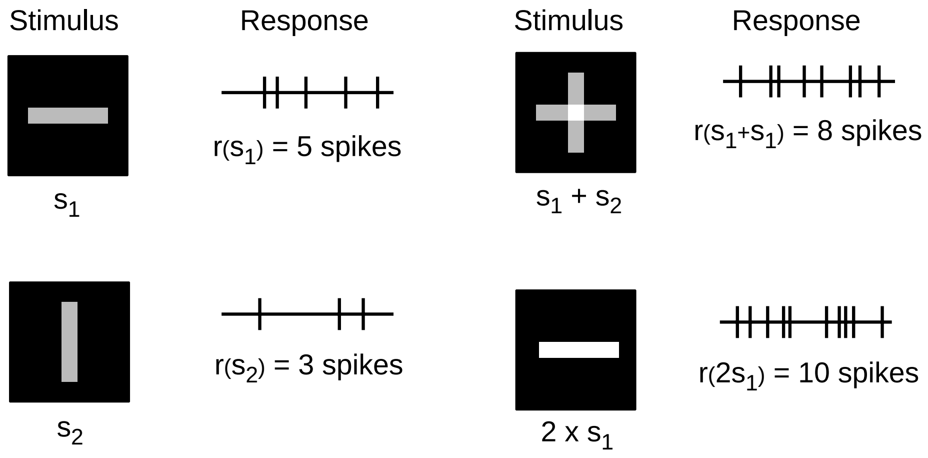


Figure 6.1: The property of superposition.

where $c_1, c_2, \ldots, c_N$ are scaling factors. Note that since the scaling factors are continuous, we have an infinite number of possible stimuli. If the neuron is truly linear, we can predict the neurons response to *any* stimulus, by simply measuring the neuron's response to each of the N special stimuli, presented one at a time. From equations (6.1)-(6.3) we have

$$r(\mathbf{s}) = r(c_1\mathbf{s}_1 + \ldots + c_N\mathbf{s}_N) = c_1 r(\mathbf{s}_1) + \ldots + c_N r(\mathbf{s}_N) \tag{6.4}$$

Therefore, to analyze a linear system, one only has to (i) break a system into it's parts, (ii) understand each part, and (iii) recombine the results. The system is then *completely* understood. The main goal of chapter 12 will be to find out how to break a linear system into parts so that the process of recombination is as simple as possible.

Warning. Using linear as interchangeable with superposition is the most common definition of the term. However, other things are sometimes meant when using the term linear, and the existence of multiple definitions can sometimes lead to confusion. The most common confusion arises when the term linear is used to describe the fact that the relationship between two variables can be plotted using a straight line. As will be shown in problem 6.3.3, in general such a relationship does *not* satisfy superposition and hence is nonlinear by our definition above. Another use of the term linear that sometimes leads to confusion is using linear to mean "able to be put in strict order" as in the term "linear thinking."

Key concept: A system is linear if it satisfies the property of superposition.

6.2 The Linear Neuron

The bulk of these notes will focus on models using very simple model neurons. While these models ignore a great deal of complexity, they correspond pretty well to the "rough-and-ready" picture of neurons that many neuroscientists use when thinking about computation in complex neural circuits. Because of this correspondence, they can be used to explore many of the basic concepts in systems neuroscience.

In these models, the entire biological neuron, including the highly branched dendritic tree, is drastically simplified into a single "compartment," or "processing unit," represented as a circle in most diagrams (figure 6.2). The internal state of such a neuron is represented by a single number s. The neuron receives input from N other neurons. This input in turn drives changes in the internal state s. The model neuron produces action potentials at a rate that is some function of the internal state, *i.e.* output rate $r = g(s)$. Depending on the background of the author, the function g has been referred to as an **output function** (or **input/output function**), a **gain function**, or a **transfer function** (transferring inputs into outputs).

Notational Aside. The derivative of the transfer function g is known as the **gain**. The gain determines how much extra output you can get per unit of extra input, $i.e.\ gain = \Delta output/\Delta input$. (The Greek letter Δ is often used to denote "change in.") In the limit of small $\Delta input$, $\Delta output/\Delta input$ is equal to the derivative, $gain = dg/ds = g'(s)$, for small changes in the input. Thus, a "high gain" transfer function is one with a steep slope.

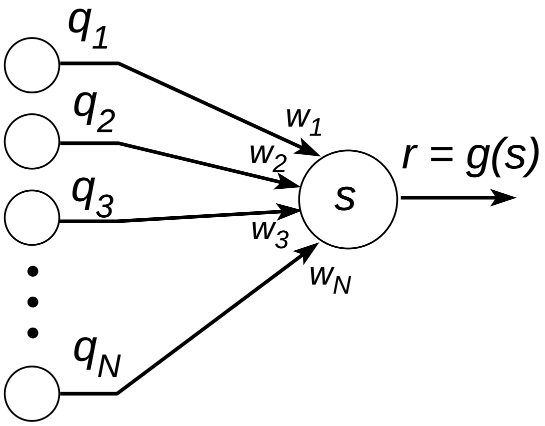


Figure 6.2: Single compartment neuron.

The simplest example of such a single compartment neuron is the **linear neuron**. The model neuron receives input from a number N of presynaptic neurons. To calculate the synaptic input current, s_j, from each presynaptic neuron j, the presynaptic firing rate q_j is multiplied by a weighting factor w_j: $s_j = w_j q_j$. w_j determines the strength or **weight** of the synaptic connection from neuron i. The total synaptic current s is just the sum of all the individual currents:

$$s = w_1 q_1 + w_2 q_2 + \ldots + w_N q_N = \sum_{j=1}^{N} w_j q_j \tag{6.5}$$

In the linear neuron the transformation from internal state to output rate is extremely simple: output firing rate is defined to be equal to s multiplied by a scaling factor g:

$$r = gs = g \sum_{j=1}^{N} w_j q_j \tag{6.6}$$

Notational Aside. Note that this notation blurs the distinction between using g to denote the input/output function ($r = g(s)$) and using it to denote a scale factor ($r = gu$). In the first case the gain is equal to $g'(s)$. In the second the gain is equal to g.

Biological Aside. The linear neuron has many non-biological simplifications. The most glaring of these is the possibility that firing rates can go negative if the neuron receives enough inhibitory (negative) input. However, we'll see that such a simple model can actually be quite useful, especially if we incorporate some simple nonlinearities explained below. In chapter ??, we'll explore more realistic model neurons.

Biological Aside. A note on units. While the appropriate units for input rates q_j and output rate r are obviously *sec*$^{-1}$ *(or Hz)*, the units for w_j depend on the biological interpretation of the neuron model. We will generally assume that w_j transforms the spike rate of presynaptic neuron j into a synaptic current, and g transforms currents into spike rates. Thus, w_j has units of *sec nA* and g has units of *nA*$^{-1}$ *sec*$^{-1}$.

Notational Aside. Being careful about keeping all your notation around can get pretty cumbersome, so computational neuroscientists often tend to be a bit sloppy (or efficient depending on your point of view). Since the range of values that a subscript can take is usually pretty clear, we often write $\sum_j w_j q_j$ for $\sum_{j=1}^N w_j q_j$. In cases with only one subscript, even that is sometimes dropped, *e.g.* $\sum w_j q_j$ means $\sum_{j=1}^N w_j q_j$. I will try to be careful to at least keep the subscript.

Since we have called this model the linear neuron, it better satisfy the property of superoposition. This is easy to check. The "stimulus" is just the pattern of input firing rates $\{q_1, \dots, q_N\}$ and the response is the output firing rate r. If we have two input patterns $\{q_1^1, \dots, q_N^1\}$ and $\{q_1^2, \dots, q_N^2\}$,

$$\begin{aligned} r(q^1 + q^2) &= g \sum_{j=1}^N w_j ({q_j}^1 + {q_j}^2) && (6.7) \\ &= g \sum_{j=1}^N w_j {q_j}^1 + g \sum_{j=1}^N w_j {q_j}^2) && (6.8) \\ &= r(q^1) + r(q^2) && (6.9) \end{aligned}$$

Similarly,

$$\begin{aligned} r(cq^1) &= g \sum_{j=1}^N w_j c {q_j}^1 && (6.10) \\ &= cg \sum_{j=1}^N w_j {q_j}^1 && (6.11) \\ &= cr(q^1) && (6.12) \end{aligned}$$

Note that the linear neuron is "doubly linear" since the transformation from input pattern $\{q_1, \dots, q_N\}$ into synaptic current s is linear, and the transformation from synaptic current s into output rate is also linear (problem 6.2.1).

Key concept: A linear neuron computes a linear transformation from input pattern to internal state (synaptic current), as well as a linear transformation from internal state to output rate.

Problems

Problem 6.2.1 (E) Show that the transformation from the vector of presynaptic firing rates to the total input s is linear. Then show that the transformation from synaptic current to output is linear.

6.3 Vector Spaces and Linear Transformations

As was mentioned in the introduction, one of the key ideas contributed by computational neuroscience is the concept of state space. The mathematics of linear operations on state spaces is known as linear algebra. Thus, to get a deeper understanding of the computational properties of networks of interconnected neurons, we'll step back a bit from the biology, and introduce the mathematical definitions and concepts basic to linear algebra.

A **vector space** is simply a collection of objects, known as **vectors**, along with the operations that are important for defining the property of superposition: (i) a method of adding two vectors; and (ii) a method of multiplying a vector by a **scalar** (or real number). Subtraction can be then be defined as addition after scalar multiplication by -1. We will focus on three different types of vector spaces.

1. Each vector is simply a list of numbers v_j:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \tag{6.13}$$

Bold lower case letters are used to denote vectors. The jth number in the list, v_j, is called the jth **component** or **element** of the vector $\mathbf{v}$. When we need to be explicit about the distinction between vectors and scalars, we will write $(\mathbf{v})_j$. Otherwise the scalar v_j is assumed to be the jth element of the vector $\mathbf{v}$. Two vectors are added by component-wise addition: $(\mathbf{u}+\mathbf{v})_j = u_j + v_j$. Scalar multiplication is defined by multiplying the scalar times each component: $(c\mathbf{v})_j = cv_j$. $\Re^N$ is often used to represent the N-dimensional vector space of real numbers ($\Re$ denotes the set of real numbers.)

Notational Aside. For reasons that will be made clear below, the list of numbers representing most vectors will be written in column form. We call these **column vectors**. Since we read from left to right, this is rather inconvenient. A **row vector** is a vector of numbers listed left to right. Luckily there is a convenient notation for the operation of switching between row and column vectors.

Definition 1 *The* **transpose** *is the operation of switching a row vector to a column vector and* vice versa*. It is represented by the symbol* $^{\mathrm{T}}$*:*

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}^{\mathrm{T}} = [v_1 \;\; v_2 \;\; \ldots \;\; v_N] \quad \mathit{and} \quad [v_1 \;\; v_2 \;\; \ldots \;\; v_N]^{\mathrm{T}} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \tag{6.14}$$

Note that $(\mathbf{v}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{v}$.

2. Each vector is an arrow in two (or three) dimensional space, whose base is placed at a special point called the **origin**. Vector addition can be defined in two ways. First, $\mathbf{u}+\mathbf{v}$ can be defined as the arrow starting at the origin and ending at the point obtained by setting the vectors tail-to-tip. Alternatively, $\mathbf{u}+\mathbf{v}$ can be defined as the diagonal of the parallelogram defined by $\mathbf{u}$ and $\mathbf{v}$ (figure 6.3a, *left*). The main reason to favor the latter definition is that

it emphasizes the fact that specifying a vector requires two points, the origin and the tip of the arrow. If we make sure that all vectors are never moved from the origin, then specifying the point at the tip uniquely specifies the vector. If we start moving vectors from the origin (as in the first definition) we have to be careful to keep track of which point is serving as the origin for each vector. Scalar multiplication simply changes the vector's length (that is why these numbers are called scalars; figure 6.3a, *middle*). Note that multiplying by a negative number flips the direction of the vector. $\mathbf{u}-\mathbf{v}$ is defined as $\mathbf{u}+(-1)\mathbf{v}$. The vector $\mathbf{u}-\mathbf{v}$ can be viewed as the vector going from the tip of $\mathbf{v}$ to the tip of $\mathbf{u}$ (figure 6.3a, *right*).

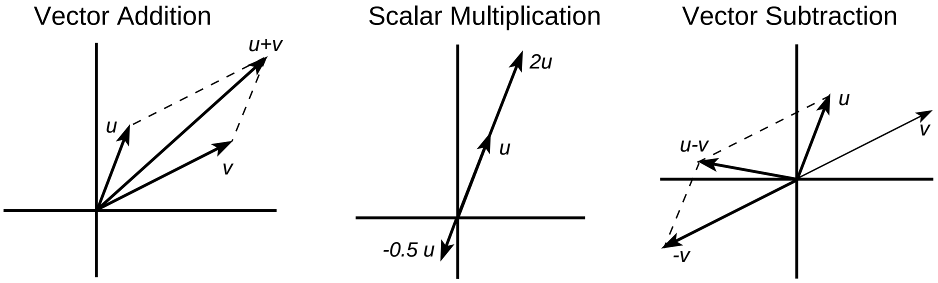


Figure 6.3: Operations on vectors.

3. Each vector is a function, *e.g.* $\sin(x)$. Vector addition and scalar multiplication are defined in the usual way: $(\sin+\cos)(x) = \sin(x) + \cos(x)$ and $(c\sin)(x) = c\sin(x)$.

Mathematical Example 6.3.1 The vector space that you are most familiar with is the one dimensional vector space of real numbers, *i.e.* lists of numbers containing only one element. Vector addition is the usual addition, and scalar multiplication is the usual multiplication. Looking at this from the geometric point of view (vector space of type 2), the vector space becomes the one dimensional number line. Note that one has to be careful in using this example since the distinction between vectors and scalars is blurred.

Mathematical Example 6.3.2 It was Descartes (1596-1650) who discovered the general equivalence between vector spaces 1 and 2, *i.e.* each arrow in a plane corresponds to a list of two numbers and *vice versa* (figure 6.4b). The operations of vector addition and scalar multiplication correspond as well. The same identification can be made in three dimensional space. This idea seems rather commonplace after being around for over 350 years, but it's really quite powerful. By identifying lists of numbers with a geometrical object, one is able to use one's geometrical intuitions to solve algebraic problems, and use algebra to solve geometric problems. Moreover, since many mathematical results are true whether the dimension is 3 or 300, one can use geometric intuitions to get insight into high dimensional problems. In fact, much of what is contained in these notes relates to getting a gut feeling of how to convert from arrows to numbers and back again. Most of the examples concern vector spaces of type 1 and 2, where the list of numbers describes the firing rate of a collection of neurons or neural populations, or the strength of the large number of synapses impinging on a given neuron. Vector spaces of type 3 will crop up now and then, often simply to illustrate a mathematical concept.

Now that we have defined vector spaces, we introduce some more terms so that we can zero in on the concept of a linear transformation. A **function** is simply a rule for taking one object (*e.g.*

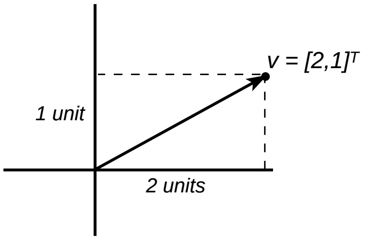


Figure 6.4: Correspondence between vectors-as-arrows and vectors-as-numbers.

a vector) as input and producing another object as output. For the function f, we write $f(x) = y$, and say that "y is a function of x." Functions are sometimes called **mappings**, since the function tells one how to "map x on to y." Similarly, functions are sometimes called **transformations** since they "transform x into y." The vector space to which x belongs is called the **domain**, and the space to which y belongs is called the **range**. In the example of linear responses in a visual neuron, the domain was the set of all stimuli, and the range was the set of responses (represented as a real number.)

Definition 2 *A mapping is a* **linear transformation** *if it satisfies the property of superposition, i.e.* $f(\mathbf{x}_1+\mathbf{x}_2) = f(\mathbf{x}_1)+f(\mathbf{x}_2)$ *and* $f(c\mathbf{x}_1) = cf(x_1)$*. Both conditions are contained in the expression* $f(b\mathbf{x}_1 + c\mathbf{x}_2) = bf(\mathbf{x}_1) + cf(\mathbf{x}_2)$*.*

Mathematical Example 6.3.3 A simple one-dimensional example of a linear transformation is $y = f(x) = mx$. It is trivial to check that this function satisfies superposition:

$$f(bx_1 + cx_2) = m(bx_1 + dx_2) = bmx_1 + cm(x_2) = bf(x_1) + cf(x_2) \tag{6.15}$$

Plotting x vs. y, we see that the graph of this function is a line (figure linfunctionfig). Note that the converse is *not* true, *i.e.* a function whose graph is a line is *not* necessarily linear (see problem 6.3.3). One can show (problem 6.3.4) that every linear map from a one dimensional range to a one dimensional range has the form $y = mx$ for some constant m.

Mathematical Example 6.3.4 Another example of a linear function is the operation of taking derivatives of functions. The derivative is a rule for taking a function f and mapping it onto a new function $f' = \frac{df}{dx}$. It follows from the definition of the derivative that it is linear (problem 6.3.2). The linearity of the derivative will be important when we address dynamical systems in chapter 12.

Problems

Problem 6.3.1 (E) Use figure 6.4a to practice making Descartes' equivalence, *i.e.* vector operations applied to vectors-as-arrows and vectors-as-number-lists are equivalent. Really think about the translation between numbers and arrows. In section **??**, we'll generalize this process and it won't be quite so trivial, so think hard about what it means to have a coordinate system.

Problem 6.3.2 (E) Recall that the derivative of a function f is defined as $\frac{df}{dx}(x) = \lim_{dx\to 0} \frac{f(x+dx)-f(x)}{dx}$. Use this definition to show that the derivative operation is linear, *i.e.* $\frac{d(bf+cg)}{dx} = b\frac{df}{dx} + c\frac{dg}{dx}$.

Problem 6.3.3 Recall that every function whose graph is a line can be written $y = mx + b$. m is the slope of the line, and b is it's y-intercept. Using the definition of the property of superposition, show that $y = mx + b$ is linear function only when $b = 0$.

Problem 6.3.4 Prove that if x and y are one-dimensional vectors, every linear function $\mathbf{y} = f(\mathbf{x})$ can be written the form $y = mx$ for some constant m. *Hint: start by looking at* $f(1)$.

6.4 The Dot Product

The input to the linear neuron was calculated by multiplying the corresponding elements of the presynaptic firing rate vector q and the synaptic weight vector $\mathbf{w}$, and then adding: $s = \sum_j w_j q_j$. This gives a linear transformation from the vector of input activities $\mathbf{q}$ to the total input s. This linear transformation was followed by another linear transformation, *i.e.* the transformation from input to output, $r = gs$. The result of concatenating two linear transformations is always linear (problem 6.4.5). This section will present a geometric picture for understanding the first transformation. In the next section, we will examine the ramifications of considering some simple nonlinearities in the input/output function.

As usual, we will need some definitions.

Definition 3 *The* **dot product** *of two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *is defined as follows:* $\mathbf{u} \cdot \mathbf{v} = \sum_j u_j v_j$. *(The terminology "dot product" arises directly from this notation.) The dot product is a special case of something known as an* **inner product***, and is sometimes written* $\langle \mathbf{u}, \mathbf{v} \rangle$ *or* $\langle \mathbf{u} | \mathbf{v} \rangle$.

From the definition it is easy to show that the dot product is linear in each of its arguments, *i.e.* $\mathbf{u} \cdot (c\mathbf{v}_1 + d\mathbf{v}_2) = c\mathbf{u} \cdot \mathbf{v}_1 + d\mathbf{u} \cdot \mathbf{v}_2$ and $(c\mathbf{u}_1 + d\mathbf{u}_2) \cdot \mathbf{v} = c\mathbf{u}_1 \cdot \mathbf{v} + d\mathbf{u}_2 \cdot \mathbf{v}$ (problem 6.4.1). But the easiest way to really understand the dot product is to connect its algebraic definition to the geometry of vector spaces.

Definition 4 *The* **length***, absolute value, or* **norm** *of a vector* $\mathbf{v}$ *is given by* $|\mathbf{v}| = \sqrt{\sum_j v_j^2} = \sqrt{\mathbf{v} \cdot \mathbf{v}}$. *This is just the standard Euclidean length of a vector viewed as an arrow in space, i.e.* $|\mathbf{v}|$ *equals the distance from the origin to the tip.*

Definition 5 *The* **distance** *between two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *is equal to the length* $|\mathbf{u} - \mathbf{v}|$. *This is just the Euclidean distance between the tips of the two vectors (see figure 6.3a,* right*).*

Definition 6 *A* **unit vector** *is a vector whose length is 1. By the above definition, the unit vector in the direction of* $\mathbf{v}$ *is* $\mathbf{v}/|\mathbf{v}|$ *(problem 6.4.3). We will use the (nonstandard) notation* $\vec{\mathbf{v}} = \mathbf{v}/|\mathbf{v}|$.

Given the relationship between the dot product and vector length, a trigonometry fact can be used to show that $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}|\ |\mathbf{v}| \cos(\theta)$ where θ is the angle between the two vectors (problem 6.4.6). Similar reasoning shows that $(\vec{\mathbf{u}} \cdot \mathbf{v})\vec{\mathbf{u}}$ is the **projection** of $\mathbf{v}$ onto $\mathbf{u}$ (see figure 6.5a). $\mathbf{u} \cdot \mathbf{v}$ is equal to the length of the projection of $\mathbf{v}$ onto $\mathbf{u}$ times the length of $\mathbf{u}$. Similarly, $\mathbf{u} \cdot \mathbf{v}$ is equal to the length of the projection of $\mathbf{u}$ onto $\mathbf{v}$ times the length of $\mathbf{v}$. The dot product also allows us to determine when two vectors are perpendicular:

Definition 7 *Two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *are* **orthogonal** *when* $\mathbf{u} \cdot \mathbf{v} = 0$. *Orthogonality is equivalent to the notion of being* **perpendicular** *since* $\mathbf{u} \cdot \mathbf{v} = 0$ *exactly when the angle between* $\mathbf{u}$ *and* $\mathbf{v}$ *is* 90^o.

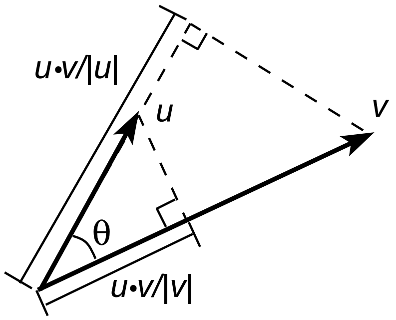


Figure 6.5: Geometric interpretation of the dot product.

With our new notation we can write the output of the linear neuron in vector notation,

$$r = g\ \mathbf{w} \cdot \mathbf{q} \tag{6.16}$$

where $\mathbf{w}$ is the vector of synaptic weightings and $\mathbf{q}$ is the vector of presynaptic firing rates. Given our geometrical interpretation of the dot product, the total input to a linear-nonlinear neuron is proportional to the length of the projection of $\mathbf{q}$ onto the weight vector $\mathbf{w}$. Thus, in a rough sense, these neurons respond to the "similarity" or "match" between the pattern of presynaptic activity $\mathbf{q}$ and the pattern of synaptic strengths $\mathbf{w}$. In fact, if we "normalize" all input vectors so that they have the same length ($|q| = 1$), $\mathbf{w} \cdot \mathbf{q}$ is proportional to the cosine of the angle between $\mathbf{w}$ and $\mathbf{q}$. Under these circumstances, the statement that the distance between $\mathbf{w}$ and $\mathbf{q}$ is less than a given radius is equivalent to the statement that the dot product is greater than some threshold value ψ (figure 6.6b).

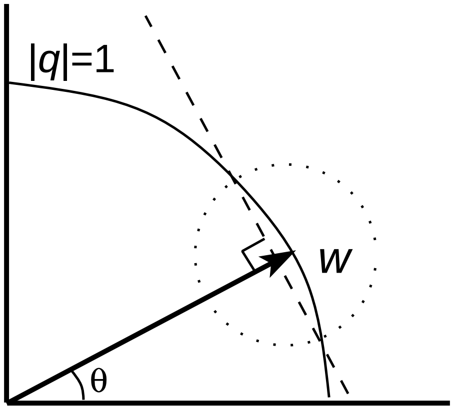


Figure 6.6: The dot product as similarity measure..

Notational Aside. For the remainder of this chapter and beyond we will set the gain $g = 1$, so that r is simply equal to the summed input. This can be viewed as a change of units that will simplify the formulas without changing any of the results. Alternatively, we can assume that g has been absorbed into the weight matrix ($\mathbf{w}_j^{new} = g\mathbf{w}_j^{old}$) so that $\mathbf{w}_j$ describes how the presynaptic firing rate gets transformed directly into postsynaptic spike rate.

One must be very careful when thinking about the selectivity resulting from taking dot products:

Warning. A pattern of strong activity in a direction not closely aligned to a neuron's weight vector can give rise to the same amount of input as a pattern of weak activity well matched to the weight vector.

Mathematically, this is just a restatement of the fact that, for fixed $\mathbf{w}$ the dot product $\mathbf{q} \cdot \mathbf{w} = \cos(\theta)|\mathbf{q}|\ |\mathbf{w}|$ depends *both* on θ *and* $|\mathbf{q}|$. One way to think about the difference between the dot product and Euclidean distance is that Euclidean distance only cares about the end point of the vector, whereas the dot product refers back to to the origin. Another way to think about it is that the dot product is most naturally expressed in polar coordinates (as a radius and angle). Given this fact, one has to worry about controlling the radius, *i.e.* the "size" of the presynaptic activity vector.

Now let's go back and think about our warning in more biological terms. Neurons often have a large number of synapses (1000-10,000 in many areas of the brain). It wouldn't be unreasonable to say that the neuron is "tuned" to detect activity in the pool of neurons strongly connected to it. The problem is that a given level of activity may be a result of either (i) moderate firing rates in a large number of these presynaptic neurons, or (ii) high firing rate in just a few of them. Looking at the output of such a neuron, one couldn't tell whether a weak version of the neuron's optimal stimulus was present, or whether a strong version of a suboptimal stimulus was present. This ambiguity is a direct consequence of the assumption that somehow neurons "integrate" or "add up" their synaptic input.

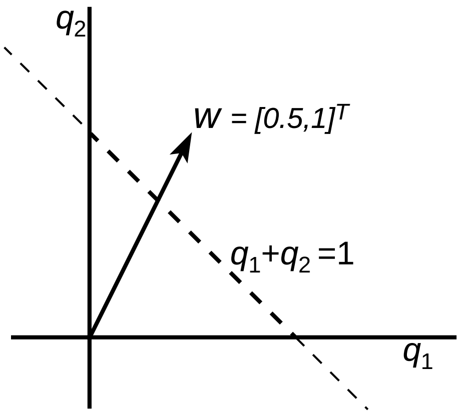


Figure 6.7: Finding the vector best matched to the weight vector $\mathbf{w}$.

Biological Example 6.4.1 As an example of the danger in thinking about dot products as a match criterion, let's address the question "what is the activity vector $\mathbf{q}$ that is best matched to the weight vector $\mathbf{w} = [0.5, 1]^{\mathrm{T}}$?" If you think about this question for a bit, you should see that the question is not well posed. If there was such a best matched vector $\mathbf{q}$, then we could just make $\mathbf{q}$ longer and we'd have even a better match (increasing $|\mathbf{q}|$ increases $\mathbf{q}\cdot\mathbf{w} = \cos(\theta)|\mathbf{q}|\ |\mathbf{w}|$). So let's add a constraint on the total amount of presynaptic activity. Suppose we say that the sum of the components $\sum_j q_j = 1$. It's easy to see that that's not good enough. Focusing on the two dimensional example, all input patterns that have total activity equal to 1 can be written $\mathbf{q} = [p, 1-p]^{\mathrm{T}}$ for some p. (Just let the first element be p, then the second has to be $1-p$ to give a total of 1.) But then $\mathbf{q}\cdot\mathbf{w} = p/2 + (1-p) = 1 - p/2$. But then as p gets to be a bigger and bigger negative number, $\mathbf{q}\cdot\mathbf{w}$ increases without bound. To get rid of such solutions with negative components, suppose we also constrain all the q_j's to be positive. Under these retrictions, we can give a concrete answer to the question of what input vector gives the best match to the weight vector $\mathbf{w} = [0.5, 1]^{\mathrm{T}}$: $\mathbf{q} = [0, 1]^{\mathrm{T}}$.

To geometrical way to see why this is the best match is shown in figure 6.7c. The dotted line shows all vectors $\mathbf{q}$ with $q_1 + q_2 = 1$. If we follow that line up and to the left we get a bigger and bigger projection onto $\mathbf{w}$. That is our first solution $\mathbf{q} = [p, 1-p]^{\mathrm{T}}$. Constraining things so that $q_j > 0$, restricts the input vectors to be in the upper right quadrant. The two constraints together restrict the inputs to the bold portion of the dashed line. It is easy to see that the end point $= [0, 1]^{\mathrm{T}}$ gives the biggest projection onto $\mathbf{w}$. This result generalizes to higher dimensions, *i.e.* given a that the input vector is positive ($q_j > 0$) and that the sum of presynaptic activity is constrained, the input vector that has the largest input to a linear neuron is one where activity is concentrated in the presynaptic neuron that has the strongest synapse (problem 6.4.7).

This example points out one of the key problems with simple model neurons. If the total input is calculated using the dot product and activity patterns are restricted to have a fixed *sum* of activity, then the optimal activity vector is not in the direction of the weight vector. To have the weight and optimal activity vector matched, we need to do something like constrain the size of the activity vector, *i.e.* constrain the sum of the *squares* of the presynaptic activities (see figure 6.6c). The problem here is that while extracting the sum of activities is easy to do biologically (problem 6.4.4), extracting the sum of the squares of activities may not be. We'll talk more about issues of normalization in chapter **??**.

Problems

Problem 6.4.1 (E) Show that the dot product is linear in each of its arguments, *i.e.* $\mathbf{u}\cdot(c\mathbf{v}_1 + d\mathbf{v}_2) = c\mathbf{u}\cdot\mathbf{v}_1 + d\mathbf{u}\cdot\mathbf{v}_2$ and $(c\mathbf{u}_1 + d\mathbf{u}_2)\cdot\mathbf{v} = c\mathbf{u}_1\cdot\mathbf{v} + d\mathbf{u}_2\cdot\mathbf{v}$.

Problem 6.4.2 (E) A. Show that $r = g\,[s-\psi]^+$ is a *nonlinear* operation. (A simple counter example will do.) B. Show that $r = g\,[s-\psi]^+$ is nonlinear even in the range $s > \psi$.

Problem 6.4.3 (E) Show that the vector $\mathbf{v}/|\mathbf{v}|$ is a unit vector.

Problem 6.4.4 (E) Construct a linear neuron whose output is equal to the sum of the activities in it's input neurons.

Problem 6.4.5 Show that the composition of two linear maps is linear, *i.e.* suppose that f is linear and that g is linear and show that the function defined by $f(g(x))$ is linear.

Problem 6.4.6 Show that $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}|\ |\mathbf{v}| \cos(\theta)$ where θ is the angle between the two vectors. *Hint:* $\cos(\theta_1 - \theta_2) = \cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2)$

Problem 6.4.7 Given a that the input vector is positive ($q_j > 0$) and that the sum of presynaptic activity is constrained, show that the input vector that has the largest input to a linear neuron is one where activity is concentrated in the presynaptic neuron that has a strongest synapse. *Hint: assume a pattern of activity that satisfies the constraint, but where more than one input unit is active. Then show that a slight change in the activity pattern can give even more input.*

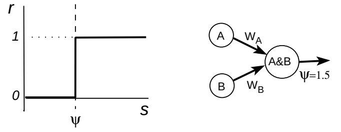
6.5 Linear-Nonlinear Neurons

We now consider the geometric interpretation of the computations performed by neurons with input/output functions $g(s) = g(\mathbf{w} \cdot \mathbf{q})$ that are nonlinear. The process of converting presynaptic rates $\mathbf{q}$ to total input s is still linear ($s = \mathbf{w} \cdot \mathbf{q}$), so I'll call them **linear-nonlinear neurons**.

6.5.1 The McCulloch-Pitts Neuron

A linear-nonlinear neuron that has a storied history in the field of computational neuroscience is the **binary neuron**, also known as the **McCulloch-Pitts neuron** since it was introduced by these two authors in an important paper published in 1943. Inspired by the all-or-none nature of action potential generation, they proposed that during each 1-2 msec time bin, a neuron would have output value 1 (emit a spike) if the summed input was greater than some threshold value, and be silent (have output value 0) otherwise. In our notation, the input/output function $g(s)$ is a step function (figure 6.8, *left*). McCulloch and Pitts asked the question whether networks of these simple neurons could be constructed to compute any arbitrary logical operation on a set of inputs. From this point of view, the output value 1 corresponds to true, and 0 corresponds to false. So if one wanted to make a network that would compute the truth value of the statement "A and B are true," one could let one input neuron represent the truth of A and another the truth of B. These could be connected to a single output unit with strength 1. If the output unit had a threshold $\psi = 1.5$, the output unit would signal "true" only if both A and B were true. McCulloch and Pitts showed that arbitrary logical operations could be performed by networks of these neurons, as long as the weights were set properly.

Biological Aside. Problems with interpretation. MORE.



A	B	C
F	F	F
T	F	F
F	T	F
T	T	T

Figure 6.8: The McCulloch-Pitts neuron.

Another common use for networks of McCulloch-Pitts neurons is pattern classification. Suppose that a neuron experiences a range of differnt types of activity patterns across its inputs, and some of these belong to category A, while others do not. This raises the following question: can connection strengths w_i and threshold ψ be found such that the neuron's output is equal to 1 whenever it is shown a pattern that belongs to A and the output is 0 whenever the input is not in A? In the early 60s, ?? published an algorithm, the **perceptron learning rule**, for setting the weights and threshold so that the problem was solved for all categories that were **linearly seperable**, *i.e.* in the space of inputs a line could be drawn so that all the patterns belonging to category A fell on one side of the line, while all the patterns not in category A fell on the other (figure 6.9b). Note that a single McCulloch-Pitts neuron can not be arranged to separate the B's and C's, *i.e.* these clusters are not linearly separable. [Have to look up more of the history of the perceptron.]

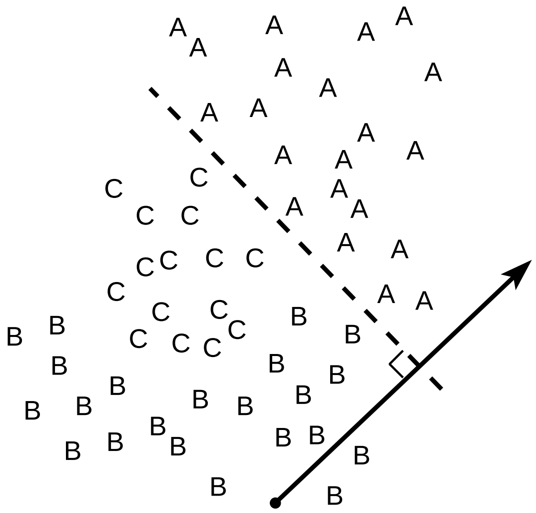


Figure 6.9: Linear separability in a category producing neuron.

6.5.2 The Sigmoid Neuron

One of the most widely used linear-nonlinear neurons is the **sigmoid** neuron (figure 6.10, *upper right*). The term sigmoid comes from the fact that the curve looks somewhat s-shaped and sigma is the Greek letter for S (OK so you have to use your immagination). Since it takes a linear input and squeezes it down to fit between zero and one, this kind of input/output function is sometimes called a **squashing function**. The sigmoid neuron can be seen as a compromise neuron that retains the ability to output a continuous range of output rates, but output rates are always positive and there is a upper limit to the allowed spike rates.

While most of the time it really doesn't matter what the exact shape of the squashing function is, there are two special functions that I will point out. The first is the **piecewise linear** input/output function (figure 6.10, *lower right*). As long as the neuron doesn't cross the "kink" in the transfer function, it is a linear neuron. As such, linear analysis techniques can be applied in piecemeal fashion to networks made up of such neurons.

Another particular function that gets used a lot is the logistic function, $r = 1/(1 + e^{(-s/T)})$. Using this function, analogies can be drawn between neural networks and statistical mechanics. The input s is viewed as the energy difference between the active and inactive state, and the rate r is the probability of the neuron being in the active state. As s increases, the probability that the neuron is in the active state approaches 1. The parameter T is analogous to temparature. When the temperature is high (T is large), random perturbations can knock the neuron back and forth between the active and inactive states, even if there is a signficant energy difference between the states. Thus, squashing functions show a gradual increase in activity (low gain) when the paramter T is large. At small temparatures (T small), even if s is only slightly different from 0, $e^{(-s/T)}$ will be near 0 or near infinitity. So as T gets smaller and smaller, the sigmoid neuron gets more and more similar to a binary neuron (high gain).

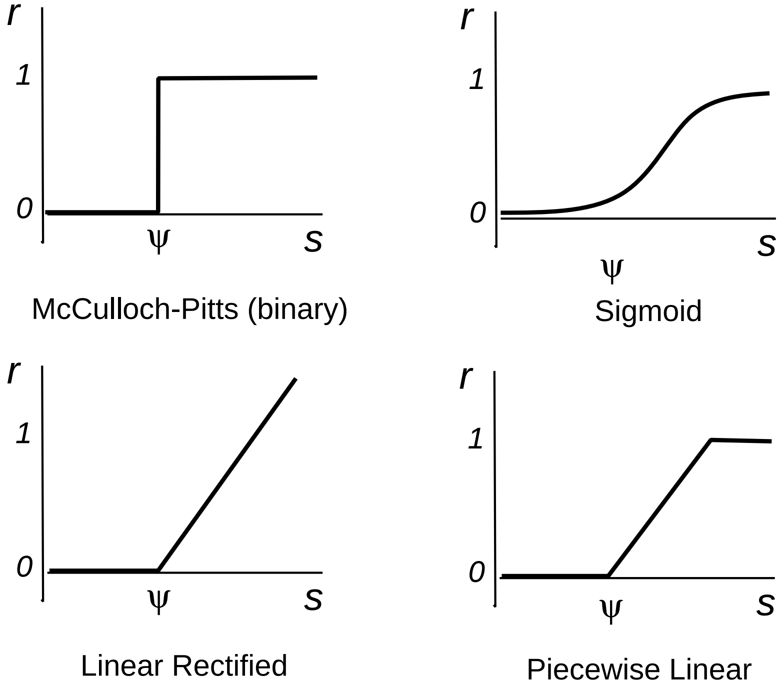


Figure 6.10: Four types of squashing function: McCulloch-Pitts or binary *(upper left)*; sigmoid *(upper right)*; linear-rectified *(lower left)*; piecewise linear *(lower right)*.

6.5.3 The Linear-Rectified Neuron

The most important linear-nonlinear neuron is the **linear-rectified neuron**. In engineering, **rectification** is the process of making an alternating current flow in only one direction. Mathematically, we define **rectification** as an operation that allows numbers to "flow" in the positive direction, but stops numbers from going negative. We can do this by comparing a number to 0 and taking the maximum: $[x]^+ = max(x, 0)$.

In real neurons, a certain amount of current is required before the membrane voltage reaches spike threshold and the neuron begins to fire action potentials. To incorporate this biological fact, the linear rectified neuron assumes that the output rate $r = 0$, until the input s reaches a threshold ψ. The output function is then "linear" after that (figure 6.10, *upper left*). We write $r = g\,[s - \psi]^+$. Note that rectification is a *nonlinear* operation (see problem 6.3.3). In fact, it is often underappreciated just how nonlinear rectification is. I would even go so far as to say that rectification due to spike threshold is the most fundamental nonlinearity in neuroscience.

6.6 Tuning Curves

So far we've introduced some simple model neurons and an abstract geometric way of evaluating their responses. This section we'll see how this linear picture relates to a more common way of presenting the responses. Most neurophysiology experiments consist of systematically manipulating some experimental variable, and recording the resulting changes in neural response. Often these results are presented in the form of a **tuning curve**. A tuning curve plots how a single response variable (usually spike rate or spike count) changes as a function of a single experimental variable. For example, one could record the number of spikes elicited in the motor cortex of a monkey, when the monkey was intructed to touch one of a number of lighted buttons. If the buttons are arranged systematically in a circle, one could make a "motor response tuning curve" by plotting spike number on the vertical axis vs. direction on the horizontal axis. An example of such a tuning curve computed by Georgeopolis and colleagues (**?**) is shown in figure **??**a.

Figure **??**a here. (Monkey tuning curve.)

How could such a tuning curve relate to our geometric picture of neurons computing dot products? Suppose for a moment that we assume that instead of getting input from many hundreds of neurons, the recorded neuron got input from just two neurons: one that responded linearly to the magnitude of the rightward motion of the monkey's arm, and one that responsed linearly to upward motion. The activity of these two neurons can be plotted in a two dimensional state space. As we ask the monkey to make motions that go around in a circle, the activity of these neurons traces out a circle in state space. (To make things simple, we'll assume that the rightward and upward neurons respond at the same rate to their optimal stimuli – rightward and upward motion respectively – and we'll express their firing rates as a fraction of this maximal response so that the circle has radius 1.) Suppose that the neuron had a connection strength $w^{right} = 2$ from the rightward neuron and a strength of $w^{up} = 1$ from the upward neuron. As the angle of motion changes, the projection along the weight vector should wax and wane smoothly, so that the tuning curve should resemble the actual data with a preferred direction of motion to the right and a little bit up (figure 6.11b).

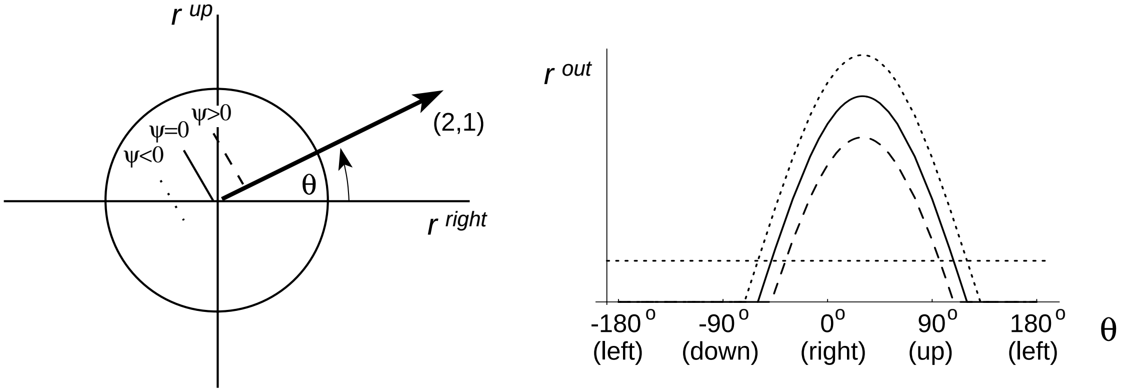


Figure 6.11: Tuning as coming from the dot product *(left)*, and shown as tuning curves *(right)*.

The width of the tuning curve is determined by the placement of threshold. For high threshold, $\psi > 0$, significant input is needed before there is any output, and the tuning curve is narrow (figure 6.11, *dashed lines*). For negative threshold, $\psi < 0$, the tuning curve is relatively broad (figure 6.11, *dotted lines*). How are we to think about a negative threshold? One clue comes from considering the case where the input is equal to zero. With a negative threshold, even no input is above threshold and the model neuron responds at a rate $r = [0 - \psi]^+ = -\psi = |\psi|$ (remember $\psi < 0$). These somewhat strange results make a bit more sense if we divide the input up into the part of the input that changes with the stimulus, I^{stim}, and the part that doesn't, $I^{background}$. Then $r = \left[I^{stim} + I^{background} - \psi\right]^+$. But since both $I^{background}$ and ψ do not change with the stimulus, they can be combined into an *effective threshold*, $\psi^{eff} = \psi - I^{background}$, so that $r = \left[I^{stim} - \psi^{eff}\right]^{+}$. A negative (effective) threshold can therefore be naturally interpreted as a neuron receiving a enough input in the unstimulated condition to produce a non-zero background firing rate. The background firing rate for the $\psi < 0$ condition in figure 6.11 is given by the horizontal dotted line.

The geometric picture gives a qualitative picture of our tuning curves. But using our simplifying assumptions, we can actually write down some formulas. Since the magnitude of the rightward motion depends on the cosine of the angle θ (measured from the horizontal), the rightward neuron's response is given by $r^{right}(\theta) = \cos(\theta)$. Similarly, $r^{up}(\theta) = \sin(\theta)$. Then, assuming a linear rectified model, $r = [2\cos(\theta) + 1\sin(\theta) - \psi]^{+} = \left[\sqrt{5}\cos(\theta + 26.6^{o}) - \psi\right]^{+}$. Thus, inputs that are a linear function of position naturally give cosine shaped tuning curves in response to stimuli described by a circular variable.

Figure **??**c here. Cricket and leech tuning curves.

Figure **??**c shows tuning curves as a function of a circular variable recorded from two completely different systems. The top plot shows a tuning curve recorded from a neuron in the terminal ganglion (a concentration of neurons) of a cricket. The stimulus parameter was the horizontal angle (relative to the animal's body) from which a (controlled) puff of wind was blown. Crickets have two appendages called cerci that stick out from the back of the animal that are covered with approximately XX thousand filiform hairs. Wind currents deflect these hairs and sensory receptors at their base detect this deflection and send signals into the terminal ganglion. There the neuron whose response shown in figure 6.11c integrates this information and sends the signal up toward the animal's head where it can trigger an escape response. The bottom plot shows a tuning curve from a neuron in the leech. This neuron reacts to objects that touch the side of the animal's body and triggers a bending reflex away from the object. The relevant experimental variable is the angle relative to the midline at which the body was touched. Note that both tuning curves are well approximated by cosine functions.

6.6.1 Push-pull

This picture is all quite satisfying. In fact, the critical reader should be wondering at this point if things are too satisfying – our model of the inputs is way too simple. The most obvious thing is that we only have two input neurons, conveniently detecting rightward and upward. Later we'll see that this really isn't that much of a restriction. However, we've also modeled the these detectors as linear. That means that we've represented leftward and downward as negative activities in the rightward and upward detectors. If we're thinking of these detectors as neurons, we've got problems since rates can't go below zero.

The most common solution to this problem is to assume that neurons come in opposing pairs, and the connections from these pairs are arranged in a "push-pull" arrangement, *i.e.* if a neuron receives an excitatory connection from one neuron, it receives an inhibitory connection from the other. For example, we can recover the linear picture of figure 6.11b if we assume that our neuron receives an inhibitory input from a leftward neuron that is the same strength as the excitatory connection from the rightward neuron, and we assume a similar push-pull arrangement for an upward and downward neuron.

There is certainly plenty of circumstantial evidence for push-pull arrangements. In the visual system. Retinal ganglion cells come in both "ON" and "OFF" subtypes. Even color seems to be represented in paired dichotomies. [This is true for red-green. I'm not so sure for blue...] More directly, for certain types of visual cortical neurons it has been shown that in locations where bright spots elicit excitation, dark spots elicit inhibition and vice versa. In the leech system, the arrangement is a "distributed push-pull," with the four sensory receptor neurons projecting onto 25-30 interneurons which then project onto 10 premotor neurons that contract or expand muscle groups arranged around the animal's body. [I have to do some reading to gather evidence for push-pull mechanisms across more systems. In fact, this might be the beginnings of a class project for someone...]

However, there are plenty of problems with this picture. First of all, to give truly linear responses, the pairing would have to be well balanced and all the neurons in question would have to have effective thresholds very near zero, otherwise the response won't look truly linear (see problem 6.6.2). A more biological criticism of push-pull as a fundamental property of neural circuits is that patterns of excitation and inhibition in the brain look quite different, not the neat mirror imaging of excitation and inhibition predicted by push-pull. For example, projections from one brain region to another are generally either all excitatory (*e.g.* thalamus and cortex) or all inhibitory (*e.g.* certain projections in the basal ganglia and cerebellum). Push-pull could still hold, but it would require some detailed circuitry involving local interneurons. A bigger problem for push-pull is that in many brain regions, the number of excitatory and inhibitory neurons are imbalanced. For example, in the cortex excitatory neurons outnumber inhibitory neurons by about 4 to 1. Ultimately, the importance of push-pull in neural processing is an experimental question.

6.6.2 Magnitude-Invariant Tuning

To end this section we return the fundamental warning about neurons that compute with dot products: their output reflects an ambiguity between input pattern and input magnitude. The manifestation of the ambiguity in our picture of tuning curves is shown in figure **??**d. We'll keep using our simple example where θ represents an input parameter like angle. To the degree that Adrian's results apply to our example, we expect that increasing the magnitude of the stimulus should result in an increased response. Therefore, stepping up the stimulus magnitude and then stepping through the angles should trace out a circle *(dashed line)* of larger radius than changing the angle at the original stimulus level. Similarly reducing the stimulus magnitude should trace out a smaller circle *(dot-dashed line)*. The ramifications in terms of tuning curves are shown at the right. What is plotted is the total input as a function of angle for each of level of stimulus magnitude. Two levels of effective threshold are shown by the thin horizontal lines. In a linear rectified neuron, the output tuning curve is obtained by simply ignoring everything that is below threshold.

MORE.

Problems

Problem 6.6.1 What would the tuning curve of the monkey neuron look like if the rightward and upward neurons were linear-rectified, just like the output neuron? Explain your answer using both the dot product and tuning curve pictures in figure 6.11b.

Problem 6.6.2 What does the input look like from a push-pull pair in which the effective threshold was not equal to zero? For example, draw the input as a function of left-right position for a leftward-rightward pair in which the effective threshold is above or below zero.

6.6.3 The Hubel-Wiesel Model

Iceberg Effect/Contrast Invariance MORE.

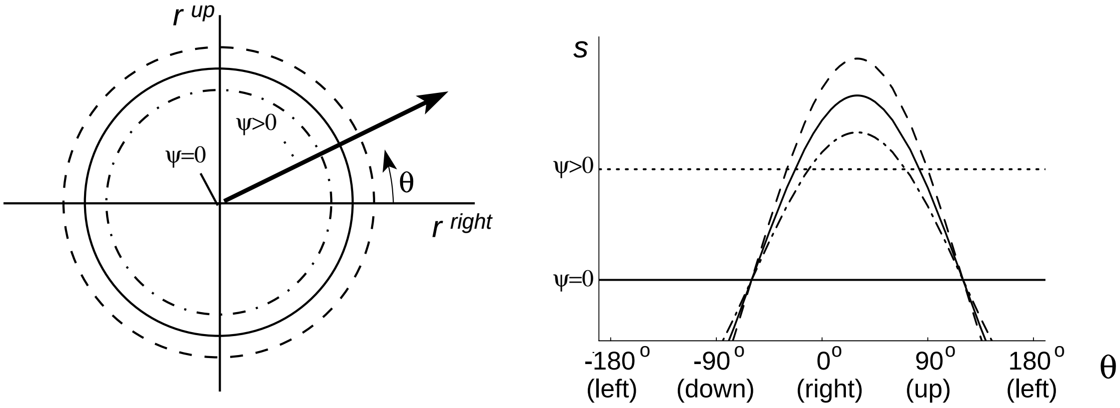


Figure 6.12: The iceberg effect.

Chapter 7

Linear and Linear-Nonlinear Networks

7.1 The Dominant Paradigm

7.2 Two Layer Networks and Linear Transformations

In the last chapter we have examined a single postsynaptic neuron receiving input from an array of N presynaptic neurons. Now we extend this picture to consider an array of P output neurons as well (figure 7.1a). Note that the two "layers" of processing units may have different numbers of neurons ($N \neq P$). We write all the strength of all possible connections between input neurons j and output neurons i in a compact, two dimensional array:

$$\mathbf{W} = \begin{bmatrix} W_{11} & \dots & W_{1j} & \dots & W_{1P} \\ \vdots & & \vdots & & \vdots \\ W_{i1} & \dots & W_{ij} & \dots & W_{iP} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ W_{N1} & \dots & W_{Nj} & \dots & W_{NP} \end{bmatrix}$$

$\mathbf{W}$ is known as a **weight matrix**. A matrix with N rows and P columns is said to be an $N \times P$ **("N by P") matrix**. Displaying the network as in the righthand side of figure 7.1a makes the correspondence between connection strength and an array of numbers easy to see.

Network Aside. For obvious reasons, these networks are commonly referred to as "two-layer" networks. However, some researchers who focus more on the patterns of weights than on patterns of activity, would refer to these networks as "single layer" networks, since they have only one layer of connection weights. This confusion is extended to multi-layer networks, with the same network called a three-layer network or a two-layer network depending on whose convention is being used. We will classify networks by the number of layers of processing units. This terminology is most common.

Notational Aside. Mathematicians often use the notation $f : \Re^N \rightarrow \Re^P$ to abbreviate the statement "the function f maps vectors in the domain $\Re^N$ into the range $\Re^P$," or more tersely "f maps $\Re^N$ into $\Re^P$." For example, single neuron models perform a mapping $f : \Re^N \rightarrow \Re^1$ (N inputs get converted to a single output).

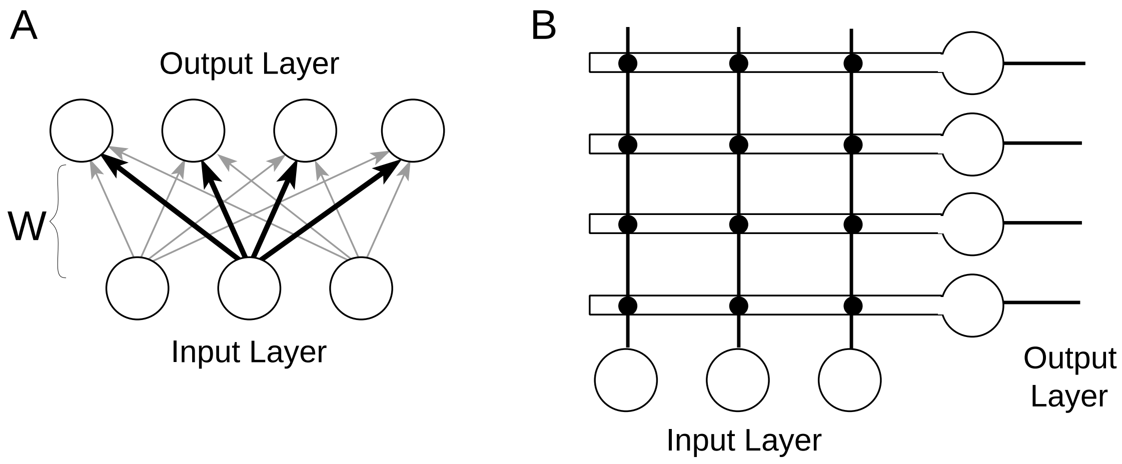


Figure 7.1: Two representations of a two-layer network.

7.2.1 The Postsynaptic (Row) Perspective

We want to define matrix multiplication in such a way that the product $\mathbf{Wq} = \mathbf{r}$ denotes a linear conversion of a pattern of inputs into a pattern of outputs. If look at the problem from the point of view of the output layer, each output neuron is acting independently. From this perspective, we can construct the output pattern element by element, calculating the activity of each output element in turn. Given the way we have written the matrix $\mathbf{W}$, *the vector of synaptic weights impinging on output neuron i is the ith row of* $\mathbf{W}$. We will denote this **row vector** by $\mathbf{W}_{i:}$. Therefore, the activity in the ith neuron, r_i, is determined from taking the dot product of the ith row of $\mathbf{W}$ and the input vector $\mathbf{q}$:

$$(\mathbf{r})_i = r_i = \mathbf{W}_{i:} \cdot \mathbf{q} = \sum_j W_{ij} q_j \tag{7.1}$$

Note that this definition can be applied to our previous case where the output layer had only one neuron. In this case, $\mathbf{W}$ is a $1 \times P$ matrix. If we let $\mathbf{w}$ be the (column) vector of weights for this neuron, we have $\mathbf{W} = \mathbf{w}^{\mathrm{T}}$. Since $r = \mathbf{w} \cdot \mathbf{q}$ using our vector notation, *and* $r = \mathbf{Wq}$ using matrix notation, $\mathbf{w} \cdot \mathbf{q} = \mathbf{w}^{\mathrm{T}}\mathbf{q}$. In other words, *given our definition of matrix multiplication, the dot product of two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *can be written as the matrix product* $\mathbf{u}^{\mathrm{T}}\mathbf{v}$. We will use both the “transpose” and “dot” notations to denote this operation.

7.2.2 The Presynaptic (Column) Perspective

Since each output element is treated separately, from the postsynaptic perspective it can be difficult to understand how presynaptic activity gives rise to a *pattern* of outputs. So now we focus on what each *presynaptic* neuron contributes to the final answer. Examining figure 7.1a reveals that the vector of synaptic weights emanating from the jth input neuron can be found in the jth column of $\mathbf{W}$. We denote this jth **column vector** $\mathbf{W}_{:j}$. By writing out the sum

$$(\mathbf{r})_i = r_i = \sum_j W_{ij} q_j = \sum_j q_j (\mathbf{W}_{:j})_i \tag{7.2}$$

we see that

$$\mathbf{r} = \sum_j q_j \mathbf{W}_{:j} \tag{7.3}$$

In other words, the final output pattern $\mathbf{r}$ is the sum of the column vectors $\mathbf{W}_{:j}$, weighted by the presynaptic activity levels q_j. That is, each presynaptic neuron drives the output toward it's vector of outgoing weights $\mathbf{W}_{:j}$, with a strength that is proportional to its activity level q_j.

Example 7.2.1 Consider the simple example where

$$\mathbf{W} = \begin{bmatrix} 6 & 0 \\ 0 & 3 \\ 2 & 4 \end{bmatrix}$$

The fact that $\mathbf{W}$ has three rows and two columns indicates that the input layer has two input neurons ($\mathbf{q}$ is two dimensional) and the output layer has three neurons ($\mathbf{r}$ is three dimensional). Suppose that this network has an input activity vector $\mathbf{q} = [1.5, 2]^{\mathrm{T}}$. What does the output look like?

Looking at the problem from the perspective of the output neurons (7.2b, *left*), we have $\mathbf{r} = \mathbf{W}\mathbf{q} = [\mathbf{W}_{1:} \cdot q_1, \mathbf{W}_{2:} \cdot q_1, \mathbf{W}_{3:} \cdot q_1]^{\mathrm{T}} = [9, 6, 11]^{\mathrm{T}}$. Notice that $\mathbf{W}_{3:}$ most closely matches the input vector and as a result the third output neuron is most active. $\mathbf{q}$ is closer in direction to $\mathbf{W}_{2:}$ than to $\mathbf{W}_{1:}$, but the activity level of the first output neuron is greater than that of the second. This is because the first output unit has the longest synaptic weight vector.

Looking at the problem from the perspective of the input neurons (7.2b, *right*), we have $\mathbf{r} = \mathbf{W}\mathbf{q} = 1.5\mathbf{W}_{:1} + 2\mathbf{W}_{:2} = [9, 6, 11]^{\mathrm{T}}$. The output is a mixture of the two column vectors, but is slightly closer in direction to $\mathbf{W}_{:2}$ since the second input neuron is more active. It is easy to see that letting $\mathbf{q}$ range over all possible values results in output vectors $\mathbf{r}$ that range over all vectors in the two-dimensional plane containing $\mathbf{W}_{:1}$ and $\mathbf{W}_{:2}$ (indicated by the dotted line in figure 7.2b, *right*). This plane is called the **image** of $\mathbf{W}$ and forms a **subspace** of the output space $\Re^3$.

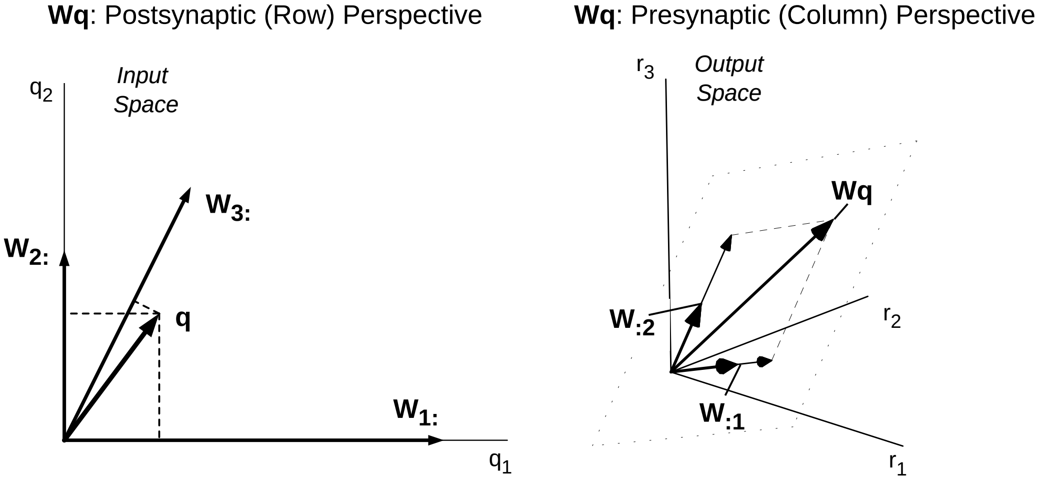
Recall that this set of all possible outputs is called the image of $\mathbf{W}$, $Im(\mathbf{W})$. If we restrict our attention only to $Im(\mathbf{W})$, the set of all vectors in $Im(\mathbf{W})$ can be viewed as its own vector space: adding two vectors in $Im(\mathbf{W})$ also yields a vector in $Im(\mathbf{W})$, and multiplying a vector by a scalar also yields a vector in $Im(\mathbf{W})$. (Mathematically we say that $Im(\mathbf{W})$ is **closed** under the operations of vector addition and scalar multiplication). $Im(\mathbf{W})$ is called a **subspace** of $\Re^3$.

Mathematical Aside. At this point it may be useful to pause and take a broad view of how far we've come toward understanding linear transformations. We started by looking at a single neuron adding up individual synaptic currents from a number of presynaptic neurons. We then made the conceptual leap to where the entire *pattern* of presynaptic activity was considered as a single object, a vector. In example 7.2.1 we get a glimpse of how an entire collection of vectors can be viewed as a single object, a subspace. Subspaces will play a key role in understanding the nature of the Hebbian learning rules that we will introduce in the next chapter.

Problems

Problem 7.2.1 (E) Show that the transformation from inputs to outputs in the two-layer network is linear.

$$\mathbf{W} = \begin{bmatrix} 6 & 0 \\ 0 & 3 \\ 2 & 4 \end{bmatrix} \qquad \mathbf{q} = [1.5, 2]^T$$



$$\mathbf{Wq} = \begin{bmatrix} 1.5 \times 6+2 \times 0 \\ 1.5 \times 0+2 \times 3 \\ 1.5 \times 2+2 \times 4 \end{bmatrix} = [\mathbf{W_{1:}\,q}, \mathbf{W_{2:}\,q}, \mathbf{W_{3:}\,q}]^T$$

$$\begin{aligned} \mathbf{Wq} &= [9, 6, 11]^T \\ &= 1.5\,[6, 0, 2]^T + 2\,[0, 3, 4]^T \\ &= 1.5\,\mathbf{W_{:1}} + 3\,\mathbf{W_{:2}} \end{aligned}$$

Figure 7.2: Postsynaptic and presynaptic perspectives on matrix multiplication.

Problem 7.2.2 (E) Practice multiplying vectors by matrices. Simply make up your own weight matrices and compute the answer for a few representative input patters. Do this two or three times. View the output from both the row and column perspective, and try to understand how the answer would change if the input pattern or weight vectors were slightly altered. The point of this exercise is not only to learn the mechanics of matrix multiplication, but to get a better intuitive grasp for what it means.

Problem 7.2.3 All planes that can be drawn in $\Re^3$ do not constitute a linear subspace from the vector space point of view. What is the key condition that distinguishes planes in $\Re^3$ that *are* linear subspaces from those that aren't.

7.3 Three-Layer Networks and Matrix Multiplication

So far we have described how to add two vectors, multiply a vector by a scalar, "multiply" two vectors using the dot product, and multiply a vector by a matrix. Like vectors, the addition of matrices is done by adding corresponding elements, *i.e.* $(\mathbf{W}^1 + \mathbf{W}^2)_{ij} = W^1_{ij} + W^2_{ij}$. Note that matrix addition is only defined if both $\mathbf{W}^1$ and $\mathbf{W}^2$ have the same size (same number of rows and same number of columns). Multiplication of a matrix by a scalar is also defined element-wise $(c\mathbf{W})_{ij} = cW_{ij}$.

Now we go on to describe how to multiply two matrices.

Network Example 7.3.1 The three-layer linear network. Suppose we add another layer of neurons to our two-layer network (figure 7.3a). Since this third layer is the final processing stage of our network we will call it the output layer. We rename the second layer the **hidden layer** since activity in this layer represents the internal workings of the network and is therefore "hidden" from an outside viewer that can only look at inputs and outputs. This network has two layers of synaptic weights, the matrix $\mathbf{W}$ of weights connecting the input and hidden layers, and the matrix $\mathbf{T}$ of weights connecting the hidden and output layers.

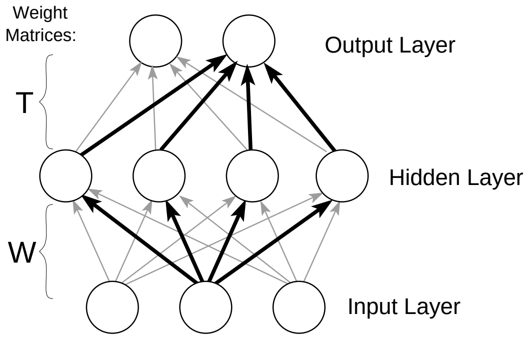


Figure 7.3: A three layer network with 3 input neurons, 4 hidden neurons, and 2 output neurons.

The first thing to notice is that the transformation from input pattern to output pattern is linear. This can be shown as follows:

$$\mathbf{T}(\mathbf{W}(b\mathbf{u}+c\mathbf{v})) = \mathbf{T}(b\mathbf{W}\mathbf{u}+c\mathbf{W}\mathbf{v}) = b\mathbf{T}(\mathbf{W}\mathbf{u})+c\mathbf{T}(\mathbf{W}\mathbf{v}) \tag{7.4}$$

We define the matrix product $\mathbf{TW}$ as the matrix that implements the transformation $\mathbf{v} \rightarrow \mathbf{T}(\mathbf{W}\mathbf{v})$. Similar to our two-layer network, we want the kjth entry of $\mathbf{TW}$ to represent how strongly input neuron j effects output neuron k. In the two-layer case, this influence can be interpreted as a synaptic strength. But now there are multiple pathways connecting these two neurons: input neuron j affects hidden layer neurons and these in turn influence output unit k. Figure 7.3a shows the multiple pathways in which input neuron 2 can influence output neuron 2 (*dark arrows*). For example suppose input neuron j's activity level increases by an amount Δq_j. This will increase the activity of hidden neuron i by an amount $W_{ij}\Delta q_j$ and this in turn will increase output neuron k by an amount $\mathbf{T}_{ki}W_{ij}\Delta q_j$. But since our network is linear, these multiple influences simply add. Therefore, the total influence of input neuron j on output neuron k is simply $\sum_i \mathbf{T}_{ki}W_{ij}\Delta q_j$.

Definition 8 *The* **matrix product** *of an $N \times Q$ dimensional matrix $\mathbf{T}$ and a $Q \times P$ dimensional matrix $\mathbf{W}$ is the $N \times P$ matrix $\mathbf{TW}$ whose kjth entry is given by $(\mathbf{TW})_{kj} = \sum_i \mathbf{T}_{ki}W_{ij}$.*

Note that $(\mathbf{TW})_{kj} = \mathbf{T}_{k:} \cdot \mathbf{W}_{:j}$. This equality gives an intuitive interpretation of matrix multiplication, the kjth entry of the product matrix $\mathbf{TW}$ represents how well the vector of weights *from* input neuron j matches the vector of weights *onto* output neuron k, where the match is measured using the dot product. We also point out that although we have added another layer of neurons, we haven't gained any information processing power: the transformation from inputs to outputs is still linear, and hence could have been implemented by a matrix of direct connections from input to output neurons. However, if the hidden units are not linear, for example if they have a threshold, then it can be shown that *any* arbitrary mapping from inputs to outputs can be implemented, as long as there are a sufficient number of neurons in the hidden layer (**?**).

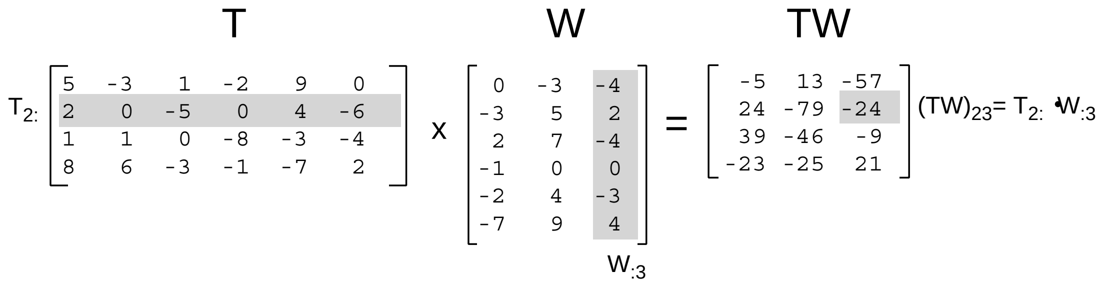


Figure 7.4: Matrix Multiplication.

7.3.1 Matrix Multiplication: The Hidden Layer View

The above perspective provides an element-by-element interpretation of the product matrix $\mathbf{TW}$, namely $(\mathbf{TW})_{kj}$ represents the net connection strength from input unit j to output unit k. There is another way to interpret the matrix $\mathbf{TW}$ that focuses on the hidden layer. The matrix $\mathbf{TW}$ represents the net effect of how a pattern at the input layer affects a pattern at the output layer. Suppose we want to examine the component of this net effect that passes through a given hidden unit i. To do so we simply ignore the other hidden units and examine the three layer network with just the one hidden unit i (see figure **??**). Using the formula for matrix multiplication we can then calculate $(\mathbf{TW})^i$, the ith"component matrix" of $\mathbf{TW}$, as $(\mathbf{TW})^i_{kj} = \mathbf{T}_{ki}\mathbf{W}_{ij}$. We can view the connections leaving hidden unit i as the column vector $\mathbf{T}_{:i}$ and the connections coming in to unit i as the row vector $\mathbf{W}_{i:}$. Then we can use matrix notation to write $(\mathbf{TW})^i = \mathbf{T}_{:i}\mathbf{W}^{\mathrm{T}}_{i:}$ (see problem **??**).

7.4 Population Coding

[NEEDS BETTER TRANSITION.]

The fact that the composition of two linear maps is a linear map allows us to shore up one of the shortcomings of the example from the last chapter that discussed cosine-shaped tuning curves (section 6.6). We drastically simplified our example by assuming only two inputs to a monkey motor neuron, one that represented upward and one that represented rightward motion. Let's remove that restriction, and suppose the motor neuron receives input from a large number of inputs, say 1000. Keeping our assumption of linearity, we assume that each of these neurons reacts linearly to rightward and upward motion. For motion in the direction θ, the amount of rightward motion is $\cos(\theta)$ and the amount of upward motion is $\sin(\theta)$. Therefore, the firing rate of the jth input neuron is $q_j = c_j^{right}\cos(\theta) + c_j^{up}\sin(\theta)$, where c_j^{right} and c_j^{up} determine the neuron's sensitivity to rightward and upward motion respectively. Letting $\mathbf{w}$ be the weight vector from these 1000 input neurons to the motor neuron in question, that neuron's output is given by

$$r = \left[\sum_j w_j \left(c_j^{right}\cos(\theta) + c_j^{up}\sin(\theta)\right) - \psi\right]^+ \tag{7.5}$$

We can rewrite this sum as

$$r \;=\; \left[\left(\sum_j w_j c_j^{right}\right)\cos(\theta) + \left(\sum_j w_j c_j^{up}\right)\sin(\theta) - \psi\right]^+ \tag{7.6}$$

$$= \quad \left[w^{right} r^{right}(\theta) + w^{up} r^{up}(\theta) - \psi\right]^{+} \tag{7.7}$$

In the last line we have viewed the amount of rightward and upward motions as effective activity levels, $r^{right}(\theta) \cong \cos(\theta)$ and $r^{up}(\theta) \cong \sin(\theta)$, and the total net influence of rightward and upward motions on the output of our cell as effective weights w^{right} and w^{up}. Thus, if we assume that the inputs are linear, we can separate the dependencies of the entire of 1000 neurons into two effective weights. The analysis showing that these inputs give rise to cosine shaped tuning curves is then identical to the argument presented in section 6.6. This was the reason behind the claim in that section that the linearity of the inputs was a more drastic assumption than the use of only two input neurons. We leave at an exercise to use vector and matrix notations to simplify the sums in equations 7.5-7.7 (problem 7.4.3).

> Mathematical Aside. The operation of matrix multiplication is so fundamental, that it is important to learn the mechanics of performing this operation, as well as getting an intuitive understanding of what it means. If we write out $\mathbf{T}$ and $\mathbf{W}$ as arrays of numbers, we see that $(\mathbf{TW})_{kj}$ is simply the dot product of the kth row of $\mathbf{T}$ with the jth column of $\mathbf{W}$ (figure 7.4b). Of course this means that the "width" of $\mathbf{T}$ must equal the "height" of $\mathbf{W}$. Also notice that multiplying a matrix times a vector is just a special case of matrix multiplication where the vector is viewed as an $N \times 1$ dimensional matrix. Even the dot product $\mathbf{u} \cdot \mathbf{v}$ is just the matrix product of the $1 \times N$ dimensional matrix $\mathbf{u}^{\mathrm{T}}$ and the $N \times 1$ dimensional matrix $\mathbf{v}$.

Problems

Problem 7.4.1 (E) Practice doing matrix multiplication. Make up two example matrices, each with 2-5 rows and columns of entries that are single digit integers (positive and negative), and multiply them. Do this enough times that you really get the hang of the mechanics (several times should be enough).

Problem 7.4.2 (E) After waiting at least one day, repeat problem 7.4.1, making up some new examples. The mechanics of matrix multiplication is one of the very few mathematical operations in this class that you should learn how to actually do it, rather than just learn what it's about.

Problem 7.4.3 Use vector and matrix notation to reformulate equations 7.6 and 7.7 into a simple vector equation. Discuss the interpretation of the matrices and vectors used in this equation. *Hint: the equation should look like the equation for a three-layer network.*

7.5 Vectors and Matrices - A Reference

(See also Batchelet reference; Batschelet (1979))

7.5.1 Basic Definitions

We will write a **vector** as a bold-faced small letter, *e.g.* $\mathbf{v}$; this denotes a *column vector.* Its elements v_j are numbers and hence are written without bold-face:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \cdots \\ v_N \end{bmatrix} \tag{7.8}$$

Here N, the number of elements, is the **dimension** of $\mathbf{v}$. The **transpose** of $\mathbf{v}$, $\mathbf{v}^{\mathrm{T}}$, is a row vector:

$$\mathbf{v}^{\mathrm{T}} = [v_1, v_2, \ldots, v_N]. \tag{7.9}$$

The transpose of a row vector, is a column vector; in particular, $(\mathbf{v}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{v}$. To keep things easier to write, we often write $\mathbf{v}$ as $\mathbf{v} = [v_1, v_2, \ldots, v_N]^{\mathrm{T}}$.

We will write a **matrix** as a bold-faced capital letter, *e.g.* $\mathbf{A}$; its elements A_{ij}, where i indicates the row and j indicates the column, are written without boldface:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \ldots & A_{1P} \\ A_{21} & A_{22} & A_{23} & \ldots & A_{2P} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & A_{N3} & \ldots & A_{NP} \end{bmatrix} \tag{7.10}$$

This is a $N \times P$ matrix, *i.e.* it has N rows and P columns. An N-dimensional vector can be regarded as an $N \times 1$ matrix, while its transpose can be regarded as a $1 \times N$ matrix. A **square matrix** is a matrix with the name number of rows as columns. The **transpose** of $\mathbf{A}$, $\mathbf{A}^{\mathrm{T}}$, is the matrix with elements $A^{\mathrm{T}}_{ij} = A_{ji}$:

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} A_{11} & A_{21} & \ldots & A_{N1} \\ A_{12} & A_{22} & \ldots & A_{N2} \\ A_{13} & A_{23} & \ldots & A_{N3} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1P} & A_{2P} & \ldots & A_{NP} \end{bmatrix} \tag{7.11}$$

Note that the transpose of a $N \times P$ matrix is an $P \times N$ matrix.

A square matrix $\mathbf{A}$ is called **symmetric** if $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$; that is, if $A_{ij} = A_{ji}$ for all i and j.

7.5.2 Special Vectors and Matrices

The N dimensional **identity matrix I** is the matrix such that $\mathbf{I}\mathbf{v} = \mathbf{v}$ for all N-vectors $\mathbf{v}$. $\mathbf{I}$ is an N dimensional square matrix with 1's along the diagonal and zeros elsewhere.

We will generally use 0 to mean any object all of whose entries are 0. It should be clear from context whether the thing that is set equal to zero is just a number, or a vector all of whose elements are 0, or a matrix all of whose elements are 0. So we abuse notation by using the same symbol 0 for all of these cases. Occaisionally we will use the symbol $\mathbf{1}$ to denote the matrix or vector all of whose entries are 1.

Matrix and vector addition and scalar multiplication

The definitions of matrix and vector addition are simple: you can only add objects of the same type and size, and things add element-wise.

Addition of two vectors:
$\mathbf{u} + \mathbf{v}$ is the vector with elements $(\mathbf{u} + \mathbf{v})_j = u_j + v_j$.
Addition of two matrices:
$\mathbf{A} + \mathbf{B}$ is the matrix with elements $(\mathbf{A} + \mathbf{B})_{ij} = A_{ij} + B_{ij}$.
Subtraction works the same way:
$(\mathbf{u} - \mathbf{v})_j = u_j - v_j$, $(\mathbf{A} - \mathbf{B})_{ij} = A_{ij} - B_{ij}$.
Scalar multiplication is also applied elementwise:
$(c\mathbf{v})_j = cv_j$, $(c\mathbf{A})_{ij} = cA_{ij}$.

The dot product

The dot product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as $\mathbf{u} \cdot \mathbf{v} = \sum_j u_j v_j$. Note that the dot product is only defined for vectors of the same length.

Matrix and vector multiplication

The multiplication of two objects (matrices or vectors) $\mathbf{A}$ and $\mathbf{B}$ to form $\mathbf{AB}$ is only defined if the number of columns of $\mathbf{A}$ (the object on the left) equals the number of rows of $\mathbf{B}$ (the object on the right). Note that this means that order matters! To form $\mathbf{AB}$, take row i of $\mathbf{A}$ ($\mathbf{A}_{i:}$), rotate it clockwise to form a column, and take its dot product with column j of $\mathbf{B}$ ($\mathbf{B}_{:j}$). That gives a single number, entry (ij) of the resulting output structure $\mathbf{AB}$. In other words, $\mathbf{AB}$ is the matrix with elements

$$(\mathbf{AB})_{ij} = \sum_k A_{ik} B_{kj} = \mathbf{A}_{i:} \dot{\mathbf{B}}_{:j} \tag{7.12}$$

General properties

Matrix multiplication is associative, *i.e.* $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$, but it is not commutative, *i.e.* $\mathbf{AB} \neq \mathbf{BA}$ even in cases where both $\mathbf{AB}$ and $\mathbf{BA}$ are defined.

Both scalar and matrix multiplication are distributive over addition, *i.e.*

$$\begin{aligned} \mathbf{A}(\mathbf{B} + \mathbf{C}) &= \mathbf{AB} + \mathbf{AC} \\ c(\mathbf{A} + \mathbf{B}) &= c\mathbf{A} + c\mathbf{B} \\ c(\mathbf{u} + \mathbf{v}) &= c\mathbf{u} + c\mathbf{v} \end{aligned}$$

Chapter 8

Linear Associations

In chapter 7 we saw how networks can implement a linear transformation of a *pattern* activity across a number of input neurons into a *pattern* of activity across a number of output neurons. The values for the synaptic weight matrices were assumed to be given. In this chapter, we examine the synaptic connection matrices arising from associative learning rules. This and then go on to discuss the decoding problem where the task is to determine the input pattern that gave rise to a given pattern of outputs. focus on the structure of linear transformations.

8.1 The Hebb Rule and LTP

Associationism has a long history in the study of the mind. Some credit Aristotle (384-322 B.C.) with making the earliest arguments that making associations between events in the world is the key to knowledge. Others trace the roots of this tradition to the philosopher David Hume (1711-1776) of "tabula rasa" fame. There was a great rise of associationism in the nineteenth century, and it lies at the heart of William James' (1842-1910) *The Principles of Psychology* (James, 1890). The formulation of associative learning that has gathered the most attention for those studying the brain was due to the psychologist Donald Hebb (??-??). In his 1949 book *The Organization of Behavior*, he made the following famous proposition:

> "When the axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such the A's efficiency, as one of the cells firing B, is increased."

This proposition has led to a number of mathematical learning rules, the simplest of which is

$$\Delta W_{ij} = \alpha r_i q_j \tag{8.1}$$

where ΔW_{ij} is the change in the weight connecting neuron j to neuron i and r_i and q_j are the firing rates of neurons i and j. α determines how much the weight changes during each association and hence the speed at which change takes place. If is often called the **learning rate**. One way to interpret this is that each time neuron j fires an action potential, the weight is increased in proportion to the activity of neuron i.

Historical Aside. The Hebb rule could easily have been known as the the James rule if computational neuroscience had been developing as rapidly at the turn of the century as it was in the 1950's. For example, James wrote

> "When the two elementary brain-processes have been active together or in immediate succession, one of them, on re-occurring, tends to propagate its excitement to the other."

In 1973, Bliss and Lomo first published evidence for a biological mechanism leading to associative change in synaptic strength between neurons (**?**). Given the centuries-long suggestion that such a mechanism could underlying learning and memory, this discovery lead to much excitement and an ongoing experimental effort to understand the cellular and molecular underpinnings of the long term increases or *potentiation* in synaptic strength that they called LTP. The associative nature of LTP is due to two important properties of a synaptic receptor called the NMDA receptor. When the neurotransmitter glutamate is released from the presynaptic terminal of many synapses in the brain, it binds to (at least) two kinds of postsynpatic receptors. Binding to the first kind of receptor, the AMPA receptor, leads to a rapid increase in current in the postsynaptic cell, possibly contributing to spiking in this cell. The action of the NMDA receptor is more complicated. At potentials below threshold, the NMDA channel is blocked by Magnesium ions. However, this block is voltage dependent, being relieved if the postsynaptic cell is depolarized to near or above threshold. Therefore, to pass current NMDA channels need to bind presynaptically released glutamate *and* to be unblocked by postsynaptic depolarization, for example by the action potential generated by the postsynaptic cell. The second important property of NMDA channels, is that part of the current that they pass is carried by calcium ions. A host of experimental results indicate that calcium then lead to a cascade of cellular events and that this eventually leads to LTP. (See for example Brown et al., 1990 for a review of Hebbian learning and LTP.)

The action of the NMDA channel is the basis for the multiplication of equation (8.1) - potentiation of synaptic strength only occurs if the presynaptic activity $q_j > 0$ (glutamate release) and postsynaptic activity $r_i > 0$ (depolarization). However, if we focus on the mathematical implications of equation (8.1), we notice one obvious drawback: given that firing rates are positive quantities, equation (8.1) implies that synaptic strengths can only increase. On the face of things, biology seems to come to the rescue: in 19XX, ?? discovered the phenomenon of long term depression (LTD). By stimulating the presynaptic neurons at a lower intensity, they were able to show that synapses can be made weaker. Subsequent experiments have led to the general hypothesis that low levels of calcium lead to LTD whereas high levels lead to LTP. This is often written as

$$\Delta W_{ij} = \alpha q_j (r_i - \phi) \tag{8.2}$$

Equation (8.2) can be interpreted as follows. Each presynaptic spike leads to the binding of glutamate at the synapse. The amount of calcium let into the cell is proportional to the postsynaptic activity, and the postsynaptic change is proportional to calcium influx, minus a threshold. Since the number of such events is proportional to q_j, we arrive at equation (8.2). Again, the constant α determines the rate of synaptic change.

The threshold in equation (8.2) models the effects of so-called **homo-synaptic LTD**, *i.e.* long term depression at the same (homo) synapse where the pre-post pairing is accomplished. In 19XX, ?? discovered the phenomenon of **heterosynaptic LTD**. This phenomenon refers to the fact that under some circumstances, inducing LTP by pre-post pairing at one synapse is accompanied by LTD at synapses that were inactive during the period of pairing. A simple way to incorporate this mechanism into a mathematical learning rule is to include a threshold on the presynaptic term:

$$\Delta W_{ij} = \alpha (q_j - \phi_{pre}) r_i \tag{8.3}$$

This makes the presynaptic term negative for synapses from inactive neurons, leading to LTD.

One can write down a Hebbian learning equation that includes both forms of LTD:

$$\Delta W_{ij} = \alpha (q_j - \phi_{pre})(r_i - \phi) \tag{8.4}$$

There are a couple of things to note about equation (8.4). First, if one sets the threshold to be the mean value of pre and postsynaptic activity, then the change in synaptic strength is related to the *covariance* of pre and post activity rather than raw pre-post *correlation*. Second, while adding the thresholds makes sense in the context of certain specific experiments, equation (8.4) leads to synaptic changes that are non-biological. For example, if there is no activity in the presynaptic or the postsynaptic neurons, equation (8.4) says that synaptic strength should increase. However, much like the linear neuron, as long as one is careful equation (8.4) can be used to illustrate basic features of Hebbian learning.

While the existence of LTD allows for the possibility for weights to go both up and down, we'll see below that it doesn't really do much to solve the problem of weights continuing to increase without bound. For now we'll just wave our hands and assume that some biological mechanism keeps the weights in a reasonable range.

The connection between associative learning, equation (8.1), and LTP was a major factor leading to the vision, popular in the 1980s, that psychological, computational, and biological approaches to understanding the brain were rapidly converging. But we've actually got a long way to go. Slight complexifications of equation (8.1) are still the dominant formulation of biological-based learning rules, but these rules have found limited application and the computational community had largely passed them by. Furthermore, even though there has been vast increase in our knowledge of the cellular mechanisms underlying LTP, clear insight into how to steer equation (8.1) and (8.2) toward biology has not been forthcoming. Finally, while associationism is still a hotly debated issue in the psychology community, the connections to biology and computation haven't been significantly strengthened in the last 20 years.

8.2 The Outer Product Rule

Returning to equation (8.1), let's see how it gives a simple account of certain phenomenon that can be characatured using our two-layer linear network. In our first example, we'll return to the turn of the 20th century and provide an formal associationist account of Pavlov's (1849-1936) famous **classical conditioning** experiments with dogs. We'll view the different layers in the network as representing different areas in the dog brain. The output layer will represent a "motor command" area of the brain, and the input layer will represent the auditory areas of the brain. We'll also assume the existence of a third set of olfactory neurons that register the smell of food (the unconditioned stimulus or US; see figure 8.1). These olfactory neurons are "hard-wired" so that every time the dog smells food this leads to a pattern of activity $\mathbf{r}^{sal}$ in the motor command area corresponding to salivation (the unconditioned reflex). Now if a bell is rung before presenting the food (the conditioned stimulus or CS), a vector $\mathbf{r}^{bell}$ of activity will be registered in the auditory area immediately followed by $\mathbf{r}^{sal}$ in the motor command area. In applying Hebb's rule (equation 8.1), we'll assume that the size of the change is appropriately controlled, and write

$$W_{ij} = r_i^{sal} r_j^{bell} \tag{8.5}$$

Note that using our notation from last chapter we can write

$$\mathbf{W} = \mathbf{r}^{sal} (\mathbf{r}^{bell})^{\mathrm{T}} \tag{8.6}$$

In chapter 6 we defined the inner product of two vectors as $\mathbf{u}^{\mathrm{T}}\mathbf{v}$. The inner product takes two $N \times 1$ dimensional vectors and yields a scalar. Now we define

Definition 9 *The* **outer product** *of two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *is the matrix* $\mathbf{u}\mathbf{v}^{\mathrm{T}}$*. If* $\mathbf{u}$ *is an* $N \times 1$ *dimensional vector and* $\mathbf{v}$ *is* $P \times 1$ *dimensional, then* $\mathbf{u}\mathbf{v}^{\mathrm{T}}$ *is an* $N \times P$ *dimensional matrix (see problem 8.2.1).*

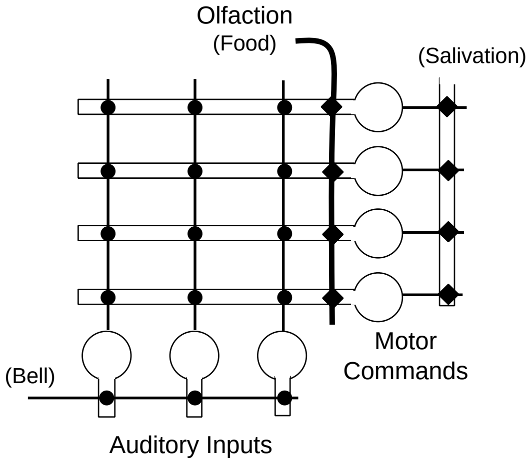


Figure 8.1: Pavlovian conditioning.

For this reason the Hebb rule (8.1) is known as an **outer product rule**.

Now what happens if we present the bell alone? In our linear network,

$$\mathbf{r}^{sal} = \mathbf{W}\mathbf{r}^{bell} = \left(\mathbf{r}^{sal}(\mathbf{r}^{bell})^{\mathrm{T}}\right)\mathbf{r}^{bell} = \mathbf{r}^{sal}\left((\mathbf{r}^{bell})^{\mathrm{T}}\mathbf{r}^{bell}\right) = \mathbf{r}^{sal}|\mathbf{r}^{bell}|^2 \tag{8.7}$$

If $\mathbf{r}^{bell}$ is appropriately normalized so that $|\mathbf{r}^{bell}|^2 = 1$, ringing the bell leads to salivation (the conditioned reflex).

For now we will leave issues of normalization aside, both in the weights and in the activities – we'll come back to them in chapter xx. Then the outer product rule can be used to give an abstract sketch of how an associational mechanism like LTP might play a role during various learning tasks.

Biological Example 8.2.1 Behaviorist Learning
To view the outer product rule as contributing to **operant conditioning**, we again view the input layer as representing some sort of sensory input and the output layer as representing motor commands that lead to behavior. But now learning is "gated" by a reinforcement signal so that without this signal no learning takes place ($\alpha = 0$). But then when the animal happens to stumble across the proper behavior, a reinforcement signal comes in and says "now print." In this way, specific stimulus-response parings can be learned, just as in classical conditioning. The main difference is that learning isn't restricted to learning behavioral responses in the animals innate repertoire. Animal trainers (and parents) have been used this type of learning for eons.

Biological Example 8.2.2 Sensory-Motor Matching
Strict behaviorists viewed the main function of the brain as taking in a sensory stimulus and responding with the motor behavior that led to the greatest reward. This overall viewpoint is still implicit in many studies of the nervous system. But it is obvious that information can flow in the opposite way. For example, when planning or performing a motor task, you don't need to wait and see what happens, you are able to generate a sensory expectation. Such expectations are crucial for obtaining the fast and fluid behavior that is necessary to survive in the world. How can these be learned? Let $\mathbf{r}^{motor}$ be a pattern of premotor activity leading to some behavior, say extending your arm. That pattern of activity will regularly be followed by a pattern of sensory input $\mathbf{r}^{sensory}$ corresponding to seeing and feeling your arm being extended. But then if connections $\mathbf{W}$ from the motor area to the sensory area are strengthened according to the outer product rule, $\mathbf{W} = \mathbf{r}^{sensory}(\mathbf{r}^{motor})^{\mathrm{T}}$, subsequent performance of $\mathbf{r}^{motor}$ will lead to an internally generated signal that carries the expectation of what the sensory input *will be*. This strategy can be used to learn a rich understanding of the capabilities of one's own body, just by randomly flailing around. Babies do quite a bit of this "motor babbling," and, as the name suggests, this is an important part of the learning complex motor behaviors such as speech. Another important functional role for motor-sensory matching, is the ability to tell the difference between sensory input that are generated by one's own actions, and those generated by external events out in the world. For example, the sensory experience one has when the entire visual world moves as a result of turning your head, is quite different than when the world moves on its own, as anybody who has experienced seasickness will tell you. In this context, the sensory signal generated from a motor command is sometimes called an **efference copy** (**?**).

The same idea used for sensory-motor matching can be used as a basic explanation of learning associations between different sensory modalities. For example, the sound of a dog barking is most often accompanied by the visual image of a dog. Hebbian learning between the auditory representation for "bark" and the visual representation for "dog" can be used to strengthen the connections between the neurons involved in these representations.

Problems

Problem 8.2.1 (E) Confirm that if $\mathbf{u}$ is an $N \times 1$ dimensional vector and $\mathbf{v}$ is $P \times 1$ dimensional, then $\mathbf{u}\mathbf{v}^{\mathrm{T}}$ is an $N \times P$ dimensional matrix.

Problem 8.2.2 (E) Confirm the equivalence of equations (8.4) and (8.12).

8.3 Multiple Memories

So far we have considered associations between a single input vector and a single output vector. What happens if we have more than one input-output pairing? To take the simplest case, suppose we have two such parings and suppose we let the connection matrix $\mathbf{W}$ just be the sum of their outer products:

$$\mathbf{W} = \mathbf{r}^1(\mathbf{q}^1)^{\mathrm{T}} + \mathbf{r}^2(\mathbf{q}^2)^{\mathrm{T}} \tag{8.8}$$

Now suppose that entries in the two input vectors $\mathbf{q}^1$ and $\mathbf{q}^2$ are uncorrelated. As we saw in chapter **??**, this is equivalent to having zero inner product, $(\mathbf{q}^1)^{\mathrm{T}}\mathbf{q}^2 = 0$. But then if we present stimulus 1, the output will be

$$\mathbf{W}\mathbf{q}^1 = \mathbf{r}^1(\mathbf{q}^1)^{\mathrm{T}}\mathbf{q}^1 + \mathbf{r}^2(\mathbf{q}^2)^{\mathrm{T}}\mathbf{q}^1 = \mathbf{r}^1|\mathbf{q}^1|^2 \tag{8.9}$$

Similarly,

$$\mathbf{W}\mathbf{q}^2 = \mathbf{r}^1(\mathbf{q}^1)^{\mathrm{T}}\mathbf{q}^2 + \mathbf{r}^2(\mathbf{q}^2)^{\mathrm{T}}\mathbf{q}^2 = \mathbf{r}^2|\mathbf{q}^2|^2 \tag{8.10}$$

It's easy to see that this argument generalizes to many memories. In the general case we write

$$\mathbf{W} = \sum_k \mathbf{r}^k(\mathbf{q}^k)^{\mathrm{T}} \tag{8.11}$$

The bottom line is that *as long as the input vectors are decorrelated (orthogonal), the outer product rule gives perfect retrieval.*

What happens if the input vectors aren't orthogonal? Well then we got problems. In fact, *dealing with correlated patterns of activity in Hebbian networks is a poorly understood and largely unsolved theoretical problem.* So even though Hebbian learning is the dominant paradigm for thinking about learning in the brain, it has found limited practical application in the computational learning community.

Mathematical Aside. Note that we can rewrite equation (8.11) using matrix multiplication notation.

8.3.1 LTD and the Outer Product Rule

In the examples presented so far, we have used the Hebb rule that only has LTP. Suppose we use the rule with pre- and/or post-synaptic thresholds for plasticity. These rules can also be expressed as an outer product by using $\mathbf{1}$to denote the vector where each element is equal to 1. Then equation (8.4) can be rewritten in vector form as

$$\Delta\mathbf{W} = \alpha(\mathbf{r} - \phi\mathbf{1})(\mathbf{q} - \phi_{pre}\mathbf{1})^{\mathrm{T}} \tag{8.12}$$

(problem 8.2.2). Adding LTD (at least in the simple way we have added it) has the effect of learning associations, not between the patterns of activities themselves, but between the patterns of activities viewed relative to a threshold.

Adding plasticity thresholds by including LTD-like effects has important ramifications for the issue of orthogonality. Two vectors that have only non-negative entries are only orthogonal if and only if the sets of neurons that are above threshold in each pattern are completely non-overlapping, *i.e.* $\mathbf{q}^1$ and $\mathbf{q}^2$ are orthogonal if and only if ${q_j}^1 > 0$ implies that ${q_j}^2 = 0$ and vice versa (problem **??**). Therefore, requiring orthogonality is equivalent to making the requirement of completely non-overlapping representations.

This restriction doesn't hold of we consider LTD thresholds. This is shown geometrically in figure 8.2. The solid vectors, $\mathbf{q}^1$ and $\mathbf{q}^2$, represent two patterns of activity, both with positive entries. The dotted vector is equal to $\phi\mathbf{1}$, the LTD threshold times the vector of all ones. The dashed vectors represent $\mathbf{q}^1 - \phi\mathbf{1}$ and $\mathbf{q}^2 - \phi\mathbf{1}$, and these vectors are orthogonal. Note that the effect of subtracting $\phi\mathbf{1}$ from $\mathbf{q}^1$ and $\mathbf{q}^2$ has the effect of "recentering" the coordinate axes (shown by the dotted lines) on the point $\phi\mathbf{1}$. This recentering leads to vectors that can have both positive and negative entries, and hence can be orthogonal without the having nonoverlapping representations.

While adding LTD threshold can remedy the most obvious restriction of requiring orthogonal memories for *learning*, the *readout* of the learning is done using the original coordinate system. That is the weight matrix

$$\mathbf{W} = \sum_k (\mathbf{r}^k - \phi\mathbf{1})(\mathbf{q}^k - \phi_{pre}\mathbf{1})^{\mathrm{T}} \tag{8.13}$$

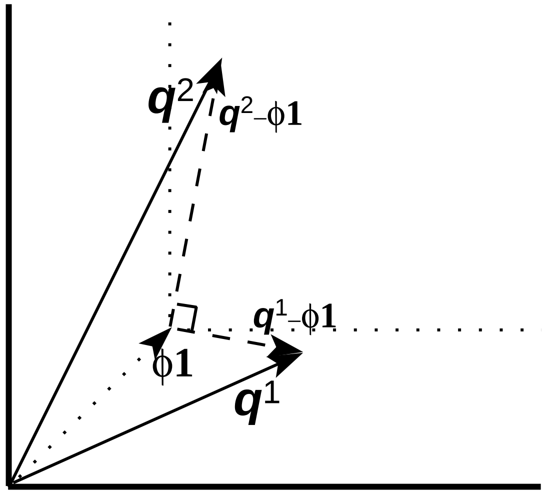


Figure 8.2: LTD thresholds can make positive vectors effectively orthogonal.

but we want to calculate the readout for a given memory, say $\mathbf{q}^1$, not $\mathbf{q}^1 - \phi_{pre}\mathbf{1}$. One way to see the effects of this mismatch in coordinate systems is to rewrite

$$\begin{aligned} \mathbf{W}\mathbf{q}^1 &= \mathbf{W}(\mathbf{q}^1 - \phi_{pre}\mathbf{1} + \phi_{pre}\mathbf{1}) && (8.14) \\ &= \mathbf{W}(\mathbf{q}^1 - \phi_{pre}\mathbf{1}) + \phi_{pre}\mathbf{W}\mathbf{1} && (8.15) \\ & && (8.16) \end{aligned}$$

Then, assuming that the memories were orthogonal in the recentered coordinates,

$$\mathbf{W}\mathbf{q}^1 = \mathbf{r}^1 - \phi\mathbf{1} + \phi_{pre}\mathbf{W}\mathbf{1} \tag{8.17}$$

Note that $\mathbf{W}\mathbf{1}$ is the vector whose ith entry is the sum of weights onto output neuron i. Therefore, if things are arranged so that ϕ_{pre} times the total weight onto a neuron is equal to the LTP threshold ϕ, this rule will give perfect retrieval. In any event, $\mathbf{W}\mathbf{q}^1$ can be transformed into the paired output $\mathbf{r}^1$ by adding the appropriate level of nonspecific input to the output layer. For example, this might be accomplished by some sort of normalization process.

Problems

Problem 8.3.1 Show that if all the entries in $\mathbf{q}^1$ and $\mathbf{q}^2$ are non-negative, then $\mathbf{q}^1$ and $\mathbf{q}^2$ are orthogonal if and only if ${q_j}^1 > 0$ implies that ${q_j}^2 = 0$ and vice versa.

8.4 Memory Subspaces

In the rest of this chapter, we'll use Hebbian learning to get a better understanding of some basic mathematical concepts. We'll revisit some of the computational problems with associational learning in chapter 12.

To get a better picture of what the outer product rule actually does, let's look at the problem from a geometric perspective. It will take a little while before the general idea of what's going on becomes clear. So we'll take a simple example, mull it over for a while, and then draw general conclusions at the end. Be patient!

Suppose we have an outer product matrix $\mathbf{W}$ built from two association pairs, $\{\mathbf{q}^1, \mathbf{r}^1\}$ and $\{\mathbf{q}^2, \mathbf{r}^2\}$. To start with the simplest case, we'll assume that all vectors are unit vectors (have length normalized to one), and that the memory vectors in both the input and output spaces are orthogonal ($\mathbf{q}^1 \cdot \mathbf{q}^2 = \mathbf{r}^1 \cdot \mathbf{r}^2 = 0$). To visualize things we'll assume that both the input layer and output layer have 3 neurons. How does the weight matrix $\mathbf{W}$ transform a general input $\mathbf{q}$? Algebraically,

$$\mathbf{W}\mathbf{q} = \left(\mathbf{r}^1(\mathbf{q}^1)^{\mathrm{T}} + \mathbf{r}^2(\mathbf{q}^2)^{\mathrm{T}}\right)\mathbf{q} = \mathbf{r}^1\left((\mathbf{q}^1)^{\mathrm{T}}\mathbf{q}\right) + \mathbf{r}^2\left((\mathbf{q}^2)^{\mathrm{T}}\mathbf{q}\right) \tag{8.18}$$

For unit vectors $\mathbf{q}^i$, $((\mathbf{q}^i)^{\mathrm{T}}\mathbf{q})$ is just the projection of $\mathbf{q}$ onto $\mathbf{q}^i$. In other words, $\mathbf{W}$ acts to (i) project each input vector onto each of the input memory vectors, and then (ii) produces an output vector that is a linear sum of output memories, weighted by the length of the projections (figure **??**A).

Now lets look at what happens to another input vector $\tilde{\mathbf{q}}$ that is made up of $\mathbf{q}$ plus a vector $\mathbf{q}_\perp$, where $\mathbf{q}_\perp$ is perpendicular to both $\mathbf{q}^1$ and $\mathbf{q}^2$ ($\mathbf{q}^1 \cdot \mathbf{q}_\perp = \mathbf{q}^1 \cdot \mathbf{q}_\perp = 0$). Then,

$$\mathbf{W}\tilde{\mathbf{q}} = \mathbf{W}\left(\mathbf{q} + \mathbf{q}_\perp\right) = \mathbf{W}\mathbf{q} + \mathbf{W}\mathbf{q}_\perp \tag{8.19}$$

But

$$\mathbf{W}\mathbf{q}_\perp = \left(\mathbf{r}^1(\mathbf{q}^1)^{\mathrm{T}} + \mathbf{r}^2(\mathbf{q}^2)^{\mathrm{T}}\right)\mathbf{q}_\perp = \mathbf{r}^1\left((\mathbf{q}^1)^{\mathrm{T}}\mathbf{q}_\perp\right) + \mathbf{r}^2\left((\mathbf{q}^2)^{\mathrm{T}}\mathbf{q}_\perp\right) = 0 \tag{8.20}$$

since we have assumed $(\mathbf{q}^1)^{\mathrm{T}}\mathbf{q}_\perp$ and $(\mathbf{q}^2)^{\mathrm{T}}\mathbf{q}_\perp$ are equal to 0. So we have $\mathbf{W}\mathbf{q} = \mathbf{W}\tilde{\mathbf{q}}$. Let $\mathbf{q}_\perp$ be the vector that starts at $\mathbf{q}$ and ends in the plane defined by $\mathbf{q}^1$ and $\mathbf{q}^2$ (see figure **??**B). We will call the set of vectors $\mathbf{q}$ such that $\mathbf{W}\mathbf{q} = 0$ the **null space** of $\mathbf{W}$, $(\mathbf{W})$. Thus, $\mathbf{q}_\perp \in (\mathbf{W})$. In general, all vectors that are in the null space of $\mathbf{W}$ are perpendicular to the memory subspace (problem 8.4.2).

We have divided the action of $\mathbf{W}$ into two different steps: *first*, project the input vector onto the input memory subspace, and *then* project onto the individual input memories to determine the relative weightings in the output memory subspace. What have we gained by this division? It seems at first that we've just added an extra projection step – after projecting onto the input memory subspace we still need to project onto the individual memories in the input space to obtain the correct output vector. To understand why we make this distinction, we redo our example with two new parings $\{\mathbf{q}_{1'}, \mathbf{r}_{1'}\}$ and $\{\mathbf{q}_{2'}, \mathbf{r}_{2'}\}$, where the new vectors are made up of combinations of the old vectors:

$$\mathbf{q}_{1'} = \left(\mathbf{q}^1 + \mathbf{q}^2\right)/\sqrt{2} \qquad \mathbf{q}_{2'} = \left(\mathbf{q}^1 - \mathbf{q}^2\right)/\sqrt{2} \tag{8.21}$$

$$\mathbf{r}_{1'} = \left(\mathbf{r}^1 + \mathbf{r}^2\right)/\sqrt{2} \qquad \mathbf{r}_{2'} = \left(\mathbf{r}^1 - \mathbf{r}^2\right)/\sqrt{2} \tag{8.22}$$

We leave it as an exercise to show that $\mathbf{q}_{1'}$ and $\mathbf{q}_{2'}$ are still unit vectors that are perpendicular to each other (problem 8.4.1). Now if we apply the outer product rule

$$\begin{aligned}
\mathbf{W}' &= \mathbf{r}_{1'}\left(\mathbf{q}_{1'}\right)^{\mathrm{T}} + \mathbf{r}_{2'}\left(\mathbf{q}_{2'}\right)^{\mathrm{T}} && (8.23)\\
&= \left(\mathbf{r}^1 + \mathbf{r}^2\right)\left(\mathbf{q}^1 + \mathbf{q}^2\right)^{\mathrm{T}}/2 + \left(\mathbf{r}^1 - \mathbf{r}^2\right)\left(\mathbf{q}^1 - \mathbf{q}^2\right)^{\mathrm{T}}/2 && (8.24)\\
&= \mathbf{r}^1(\mathbf{q}^1)^{\mathrm{T}} + \mathbf{r}^2(\mathbf{q}^2)^{\mathrm{T}} \ = \mathbf{W} && (8.25)
\end{aligned}$$

The new vector pairs give the same weight matrix as the old! *By looking at the entries in the matrix* $\mathbf{W}$*, we can't tell what the individual memories were, we can only determine information* *about the memory subspaces, both input and output.* This is why it is conceptually useful to separate the projection onto the input memory subspace into a separate step. This first step is identical for both sets of memories. But the two sets of memories lead to *different* projections within the input memory subspace and *different* recombinations in the output memory subspace, *but these calculations arrive at the same final answer* ($\mathbf{W} = \mathbf{W}'$). The underlying reason for this important fact is rather simple. The matrix $\mathbf{W}$ is determined by the statistical structure of the memories, rather than the memories themselves. Therefore, any set of memories that have the same statistical structure will lead to the same weight matrix.

Problems

Problem 8.4.1 (E) Show that $\mathbf{q}_{1'}$ and $\mathbf{q}_{2'}$ are unit vectors and that they are perpendicular to each other.

Problem 8.4.2 If $\mathbf{W}$ is obtained as a sum of outer products via Hebbian learning, show that $(\mathbf{W})$ is equal to the set of all vectors that are perpendicular to the input memory subspace.

Chapter 9

Neural Decoding

9.1 Taking the Organism's Perspective

Much of computational neuroscience research is concerned with the question of representation. How are sensory stimuli represented inside the brain? What kinds of motor representations are used to guide movement? Rudimentary forms of these questions can formulated rather easily within our simple linear-nonlinear networks. On the sensory side, we can view the input layer as containing the patterns of activity over a receptor array, or, as we saw in the last section, the input layer could be seen as directly representing abstract parameters describing the stimulus. On the motor side, it is often most appropriate to view the "input" layer as representing premotor activity (*e.g.* in motor cortex), and view the output as representing motor unit activity,the neural output given to the musculature, or perhaps even the parameters describing the resulting motion.

9.2 Probabilistic Approaches

We already encountered the problem of decoding in chapter **??**. All of the ideas introduced there (maximum likelihood, MAP, etc.) can be applied to the problem of decoding the responses generated by an entire population of neurons. Instead of a simple one dimensional variable, the response is an entire vector. As a result, the relevant probability distributions live in a high dimensional space and, in the general case, are difficult to characterize using reasonable quantities of data. However, things become much easier under the assumption of independent noise, *i.e.* the variability between neurons is uncorrelated. In this case, the likelihood of the entire vector of responses $\mathcal{P}(\mathbf{r}|s)$ is just the product of the likelihoods $\mathcal{P}(r_i|s)$ of the responses of each neuron taken individually. Mathematically we write[1]

$$\mathcal{P}(\mathbf{r}|s) = \prod_i \mathcal{P}(r_i|s) \tag{9.1}$$

Because taking the product creates complex dependencies between the individual likelihoods, one often maximizes the log of the likelihood (called the **log likelihood**) rather than the likelihood itself. The reason is that taking the log turns products into sums:

$$\log\left(\mathcal{P}(\mathbf{r}|s)\right) = \log\left(\prod_i \mathcal{P}(r_i|s)\right) = \sum_i \log\left(\mathcal{P}(r_i|s)\right) \tag{9.2}$$

Note that finding the maximum of the log likelihood is equivalent to finding the maximum of the likelihood since $\log(x)$ is a strictly increasing function of x.

---

[1] The symbol $\prod$ is used to denote products in a manner similar to the way that $\sum$ is used to denote summations.

Just as before, what the animal needs to calculate is maximum *a posteriori* (MAP) estimate of the stimulus given the response, $\mathcal{P}(s|\mathbf{r})$. Applying Bayes' rule we have that

$$\mathcal{P}(s|\mathbf{r}) = \frac{\mathcal{P}(\mathbf{r}|s)\mathcal{P}(s)}{\mathcal{P}(\mathbf{r})} \tag{9.3}$$

While finding the MAP or Bayes optimal decoding scheme for a given set of experiments is often possible, the relevance for understanding how the brain works is unclear. In contrast, we have seen in chapter 7 that it is quite easy to perform linear (or linear-nonlinear) transformations of patterns of neural activity using hardware that at least vaguely resembles the properties of biological neurons. One thing that calculating the optimum decoding *does* do however, is to set an upper bound on how good any decoding scheme *could be*, at least among the decoding schemes that are possible given the limitations of the experimental set up and the manner in which the data are quantified. We now turn away from our focus on noise and probability distribution, and address the problem of population coding problem in the context of simple networks.

9.3 Optimal Linear Maps

As usual we will begin with the case of purely linear networks, and will view both the input (stimulus) patterns and output (response) patterns as vectors. Generally, the input vector is interpreted not as a pattern of activity across an array of sensory neurons, but as a vector that parametrizes the set of stimuli presented to the system. It is important to remember that there are usually an entire range of ways in which the stimuli can be described as a vector of variables. The linear decoding schemes described below may work well for some such encodings, and not work at all for others. It is sometimes possible to turn a problem that appears complex and highly nonlinear into a much simpler problem by making the proper (nonlinear) transformation of the stimulus variables at the start.

Given this caveat we will examine the question of how to find the connection matrix that gives the "best" mapping between input vectors and output vectors. The same mathematical framework applies to whether we're focusing on encoding (mapping inputs to outputs) or decoding (mapping outputs to predicted input). We'll focus on the inverse problem, *i.e.* mapping output vectors onto (predicted) input vectors. We will begin by defining the problem and giving a few biological examples. Then the solution to the problem will be presented, along with some of the necessary mathematical formalism. Finally, we will discuss the solution in the context of the examples.

WARNING INPUT AND OUTPUT SPACE MAY SOMETIMES GET MIXED UP IN THE PRESENTATION.

In the basic version of the problem, we are given a collection of pairs of input and output vectors, $(\mathbf{u}^\alpha, \mathbf{v}^\alpha)$, where $\alpha = \{1, 2, \ldots, N\}$ is the index enumerating these pairs. The goal is to find the matrix $\mathbf{T}$, such that $\mathbf{T}\mathbf{v}^\alpha = \mathbf{u}^\alpha$. In many neural cases, it isn't possible to find a matrix $\mathbf{T}$ that maps output vectors *exactly* onto input vectors. In this case, we're interested in finding $\mathbf{W}$ that minimizes the squared error, $\sum_\alpha (\mathbf{T}\mathbf{v}^\alpha - \mathbf{u}^\alpha)^2$.

9.3.1 Example #1: Optimal Memory Retrieval

We've already encountered a version of the problem of learning a map from inputs to outputs in chapter 8. There we showed that simple correlational learning could yield an accurate mapping as long as the input vectors were orthogonal and had length equal to one. These conditions will arise naturally from the conditions that we will derive mathematically below.

9.3.2 Example #2: Linear Receptive Fields

A second group of examples come from re-examining the notion of the linear receptive field. In chapter 6 we calculated the linear receptive field as the average stimulus before a spike. This average stimulus can be a spatial pattern (*e.g.* a simple cell receptive field consisting of ON and OFF subregions), a temporal pattern (*e.g.* the pattern of motion velocity before a spike generated by the H1 neuron in the fly), or a spatiotemporal pattern (*e.g.* the full spatiotemporal receptive field of a simple cell).

WILL PRESENT/REVIEW IN CLASS

In the simple spatial example, the output vector is one-dimensional and takes on binary values (either the presence or absence of a spike). By discretizing space, the receptive field pattern can be viewed as a weight vector, *i.e.* as a list of connection strengths determining how strongly a stimulus in that spatial/temporal location is related to one-dimensional output vector. In this case, the linear receptive field can be interpreted in either the forward/coding direction or the backward/decoding direction.

In the example of the temporal receptive fields, the input and output vectors represent sequences of values indexed by time: motion velocity for the input and binary vectors representing spike trains for the output. Conceivably, the data could come from a single continuous recording, and there would be a single input vector and a single output vector. However, in practice it is assumed that the presence of a spike depends on the stimulus duug a relatively short time window before the spike. To keep the problem within our general framework, we will view the problem as finding the optimal mapping between a number of shorter length input and output vectors that are taken from the full train of data. We'll start by ignoug the effects due to the edges of such windows, and then go back an show that the solution can be applied to the general case.

Finally, the full spatiotemporal problem can be solved by fairly straightforward application of the pucíples derived for the purely spatial and temporal cases.

9.3.3 Example #3: The population vector

Our third example relates to the population vector approach, first popularized by Georgeopoulos and colleagues (). Recording the motor cortex of monkeys, they constructed motor tuning curves, a plot of the average number of spikes in the period before a pointing movement vs. the direction of that movement. (In the original experiment, the monkey was cued to make pointing movements toward one of eight location arranged in a circle around the central, "resting" location.) Many of neurons were broadly tuned, with tuning curves were well approximated by a cosine function. Thus it seemed that the monkey was using a population code to represent movement direction in the motor cortex. Moreover, Georgeopoulos and colleagues reasoned that a fairly simple scheme could be used to read out this code, namely each neuron could be associated with a vector corresponding to the peak direction of the tuning curve, and the direction of the upcoming movement could be decoded by multiplying each of these vectors by the activity of the corresponding neuron and summing using vector addition. This method of decoding was known as the population vector approach, and was shown to be fairly successful at predicting movement direction as long as data from a sufficient number of neurons was included (**?**).

The population vector approach fits easily within the linear network framework presented in chapter 7. Focusing for now on the case of movements in two dimensions (later experiments were done using movements in three dimensions), we can view target location as a two-dimensional vector $\mathbf{u}$ representing horizontal and vertical location. If we consider neurons whose tuning curves have a cosine shape (around a positive average level of activity), then the output of neuron i can be written as

$$\mathbf{v}_i = \mathbf{W}_{i:}^{\mathrm{T}} \cdot \mathbf{u} + \eta \tag{9.4}$$

where the row vector $\mathbf{W}_{i:}$ gives the direction vector for unit i, and η represents noise. Negative activity levels represent fiug rates that are below average. Equation 9.4 represents the *encoding* portion of this problem. In the decoding portion, this output vector is transformed into a *prediction* of the corresponding input vector. If this is a linear process, the encoding and decoding process can be represented as a three layer network, where the input layer represents the actual target location, the hidden layer represents the pattern of activity across a number of motor neurons, and the output layer contains the population vector which can be compared to the actual target (figure 9.1). In calculating the accuracy of any decoding scheme, we need to consider the effects of noise in the pattern of neural activity (the hidden layer).

In the population vector approach, the only thing that is taken from measuring these tuning curves is the set of direction vectors $\mathbf{W}_{i:}$. In the *decoding* phase, we start with a vector of fig rates, $\mathbf{v}$ and we want to predict the ensuing direction of movement. The population vector is calculated as

$$\mathbf{u}^{pred} = \sum_i \mathbf{v}_i \mathbf{W}_{i:}^{\mathrm{T}} \tag{9.5}$$

But since the rows of $\mathbf{W}$ are just the columns of $\mathbf{W}^{\mathrm{T}}$, equation (9.5) is can be rewritten

$$\mathbf{u}^{pred} = \mathbf{W}^{\mathrm{T}}\mathbf{v} \tag{9.6}$$

That is, the population vector approach takes $\mathbf{T} = \mathbf{W}^{\mathrm{T}}$ as the decoding transformation.

Population coding has been examined in a number of other systems. Two well-studied systems that are particularly relevant are the wind detection system of crickets and the lateral bend reflex in leeches (figures **??** and **??**). The relative simplicity of these invertebrate systems has facilitated quantitative investigations of neural coding that are difficult to perform in vertebrates.

FIGS/EXPLANATION HERE.

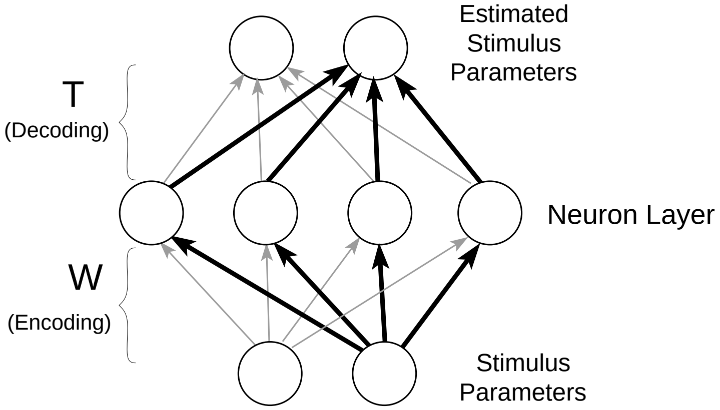


Figure 9.1: The population decoding problem viewed as a three layer network.

9.4 Formal Solution to the Linear Decoding Problem

We now turn to the mathematical description of the decoding problem and it's solution.

Recall that given a collection of input/output pairs $\{\mathbf{u}^\alpha, \mathbf{v}^\alpha\}$, we'd like to find the matrix $\mathbf{T}$ such that $\mathbf{T}\mathbf{v}^\alpha = \mathbf{u}^\alpha$. First we rewrite the problem in matrix form:

$$\mathbf{T}\mathbf{V} = \mathbf{U} \tag{9.7}$$

where $\mathbf{V}$ and $\mathbf{U}$ are the matrices where the αth column is given by the appropriate output or input vector, $\mathbf{V}$ or $\mathbf{U}$.

$$\mathbf{V} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{v}^1 & \mathbf{v}^2 & \cdots & \mathbf{v}^N \\ \vdots & \vdots & & \vdots \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^N \\ \vdots & \vdots & & \vdots \end{bmatrix} \tag{9.8}$$

Proceeding formally we can simply solve for $\mathbf{T}$:

$$\mathbf{T} = \mathbf{U}\mathbf{V}^{-1} \tag{9.9}$$

If we consider the population coding situation (and ignore noise), we have $\mathbf{v}^\alpha = \mathbf{W}\mathbf{u}^\alpha$ or $vm = \mathbf{W}\mathbf{U}$. The population decoding problem becomes one of finding the decoding matrix $\mathbf{T}$ so that $\mathbf{T}\mathbf{W}\mathbf{u} = \mathbf{u}$. In other words, $\mathbf{T}\mathbf{W} = \mathbf{I}$. In this case, we should have $\mathbf{T} = \mathbf{W}^{-1}$. In either case we need to know how to find the matrix inverse.

9.5 Existence of a Matrix Inverse

We'll start with the formal mathematical definition:

Definition 10 *Given a transformation* $f : \Re^N \to \Re^P$*, the* **inverse** *of* f *is a mapping* $f^{-1} : \Re^P \to \Re^N$*, such that* $f^{-1}(f(\mathbf{u})) = \mathbf{u}$ *and* $f(f^{-1}(\mathbf{v})) = \mathbf{v}$ *for any vectors* $\mathbf{u} \in \Re^N$ *and* $\mathbf{v} \in \Re^P$*. If such an inverse mapping exists, we say that* f *is* **invertible***.*

> **Mathematical Example 9.5.1** The simplest example of an inverse of a linear transformation is the operation of division applied to simple number , *i.e.* the inverse of the transformation $y = mx$ is the inverse transformation $x = (1/m)y$.

The most basic question one can ask is, given a mapping, does its inverse even exist? There are two basic ways in which a function can fail to be invertible. As an example of the first, consider the problem of moving your finger to a given position in space, say in the location of the period at the end of this sentence. In this case, we could imagine the input to be the patterns of motor activity controlling the position of your arm and hand, and the output to be the location of your finger. The inverse problem is to determine the motor pattern of activity that caused your finger to point to the period. It is easy to see that this is not a well-defined problem: there are many *different* patterns of activity that would lead your finger to arrive at the same location in space – your elbow could be down or to the side, your finger could be straight or bent, etc. We say that the mapping of motor activity to finger location is **many-to-one**, and, in this case, a well-defined inverse does not exist, *i.e.* there is no transformation that takes position as input and gives *the* corresponding pattern of motor activity as output. The fact that there is not a well-defined inverse plays a crucial role in many computational issues related to motor behavior. Turning the example around, we can say the following: a necessary condition for a map f to have an inverse, is for f to be **one-to-one (1-1)**, *i.e.* every output pattern has exactly one input pattern that maps onto it. Transformations that are not 1-1 are sometimes referred as being **many-to-one**.

To give an example of the second way that a function can fail to have an inverse, consider the visual system. There are approximately XX thousand axons projecting from the retina to the brain, but roughly XX neurons in the layer of the primary visual cortex that (indirectly) receives the input from these axons. That is, there are approximately 10? times as many output neurons as input neurons. That suggests that there are a lot more possible patterns of cortical activity than there are possible patterns of retinal activity. These "extra" patterns of cortical (output) activity don't have a corresponding input pattern. Thus the function mapping retinal inputs into cortical activity patterns is not invertible.

> **Mathematical Example 9.5.2** Note that there is one linear mapping of the form $y = mx$ that is not invertible, name the transformation with $m = 0$. Since this mapping takes every y onto the single value $x = 0$, it is many-to-one. Also, all values $y \neq 0$ do not have a pre-image x. Thus, multiplication by 0 fails to be invertible for both of the above reasons.

At the risk of causing some confusion, let's reinterpret the visual example starting by taking the organisms perspective. Suppose that the animal assigns each cortical state to the retinal image that is most consistent with that state. The greater number of neurons in the cortex means that there may be many internal ways to represent the same image. We say that the cortical representation is **overcomplete**. By assigning a retinal image to the "extra" cortical states, it seems as though we *can* invert the transformation from retina to cortex. What's different from the previous paragraph? The difficulty lies that in proposing an overcomplete cortical representation, each retinal state corresponds to a number of cortical states. Hence we can no longer view this system as transforming retinal images into cortical activities in a well-defined manner. Another way to say this, is that the decoding mapping from cortex to retina is not 1-1, so *it* is not invertible.

Problems

Problem 9.5.1 Show that if $\mathbf{W}$ is linear and invertible, then $\mathbf{W}^{-1}$ is also linear. *Hint: Write down the linearity condition for* $\mathbf{W}$*, and apply the transformation* $\mathbf{W}^{-1}$ *to both sides.*

9.6 The Image and Null Space

Now we define the mathematical concepts necessary to describe these two reasons for not having an inverse. The difficulty of having an output space that is "bigger" than the input space can be described using the following definition:

Definition 11 *The image, $Im(f)$, of linear transformation $f : \Re^P \rightarrow \Re^N$ is the set of all vectors in $\Re^N$ that can be reached by applying f to some vector $\mathbf{u} \in \Re^P$. Note that $Im(f)$ lies within the output space (the range) of f.*

For f to be invertible, it must be the case that $Im(f) = \Re^N$. If a mapping satisfies this condition, we use the rather ungrammatical terminology and say that "the mapping f is **onto**." (The term "onto" should be contrasted with a function that maps one space *into* another.)

To clarify the conditions that make a linear map onto, let's return to the presynaptic view of multiplying a vector time a matrix, $\mathbf{Wu} = \mathbf{v}$, that were presented in chapter 7. From the presynaptic view, the output vector is a linear combination of the columns of $\mathbf{W}$: $\mathbf{v} = \sum \mathbf{W}_{:j}\mathbf{u}_j$. From this point of view it can be seen that the image of $\mathbf{W}$, $Im(\mathbf{W})$ is all the vectors in the output space that can be written as a linear combination of the column vectors of $\mathbf{W}$. We say that $Im(\mathbf{W}) = Span\{\mathbf{W}_{:1}, \mathbf{W}_{:2}, \ldots, \mathbf{W}_{:N}\}$. The image is often call the **column space** of $\mathbf{W}$. For the mapping given by $\mathbf{W}$ to be onto, we must be able to reach every output vector using a linear combination of the column vectors of $\mathbf{W}$. Thus it is clear that the number of columns of $\mathbf{W}$ must be at least as large as the dimension of the output space. While this numbers argument yields a *necessary* condition, it is not sufficient. For example, suppose that every column of $\mathbf{W}$ was exactly the same. Then, the output space would be one-dimensional. To make things rigorous, we need the following definition:

Definition 12 *A collection of vectors* $\{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^N\}$ *is said to be* **linearly independent** *if the only way to satisfy the condition* $c_1\mathbf{v}^1 + c_2\mathbf{v}^2 + \ldots + \mathbf{c}_N\mathbf{v}^N = 0$ *is for* $c_1 = c_2 = \ldots = c_N = 0$.

One way to understand this definition is that it implies that no single vector in the collection can be written as a linear combination of the others. For example, if $\mathbf{v}^1 = c_2\mathbf{v}^2 + \ldots + \mathbf{c}_N\mathbf{v}^N$, then $-\mathbf{v}^1 + c_2\mathbf{v}^2 + \ldots + \mathbf{c}_N\mathbf{v}^N = 0$, and the vectors are not independent. A more geometric definition of linear independence is that each vector has a positive projection on a direction that is perpendicular to all the other vectors (see problem **??**). Therefore, we can say that the mapping given by $\mathbf{W}$ is onto exactly when $\mathbf{W}$ has N linearly independent column vectors.

Now let's consider a linear map $\mathbf{W}$ that is not 1-1. That means that there are at least two input vectors $\mathbf{u}^1$ and $\mathbf{u}^2$ that map onto the same output vector $\mathbf{v}$. But since $\mathbf{W}$ is linear,

$$\mathbf{W}(\mathbf{u}^1 - \mathbf{u}^2) = \mathbf{W}\mathbf{u}^1 - \mathbf{W}\mathbf{u}^2 = \mathbf{v} - \mathbf{v} = 0 \tag{9.10}$$

The key concept we need is the following:

Definition 13 *The set of all vectors* $\mathbf{u}$ *such that* $\mathbf{W}\mathbf{u} = 0$ *is called the* **null space** *of* $\mathbf{W}$, $Null(\mathbf{W})$. $Null(\mathbf{W})$ *is also called the* **kernel** *of* $\mathbf{W}$. *Note that the null space of* $\mathbf{W}$ *lies within the input space (the domain) of* $\mathbf{W}$.

The statement that $\mathbf{W}$ is 1-1 is equivalent to the statement that $Null(\mathbf{W})$ consists of a single vector, namely the 0-vector (the vector with all entries equal to 0)(problem 9.6.1). Suppose we have a non-zero vector $\mathbf{u}$ belonging to $Null(\mathbf{W})$. Then $\mathbf{W}\mathbf{u} = 0$. Taking the postsynaptic view of matrix multiplication, this means that $\mathbf{W}^{\mathrm{T}}_{i:} \cdot \mathbf{u} = 0$ for every row vector $\mathbf{W}^{\mathrm{T}}_{i:}$. The space of all vectors spanned by the rows of $\mathbf{W}$ is called the **row space** of $\mathbf{W}$. So the null space is orthogonal (perpendicular) to the row space of $\mathbf{W}$.

Now suppose we have an input vector $\mathbf{u}$. Using a perpendicular projection, we can write $\mathbf{u}$ as the sum of a vector $\tilde{\mathbf{u}}$ that lies in the row space of $\mathbf{W}$ and a vector $\mathbf{u}^{\perp}$ that lies in the null space of $\mathbf{W}$.[2] But then

$$\mathbf{W}\mathbf{u} = \mathbf{W}(\tilde{\mathbf{u}} + \mathbf{u}^{\perp}) = \mathbf{W}\tilde{\mathbf{u}} + 0 = \mathbf{W}\tilde{\mathbf{u}} \tag{9.11}$$

In other words, all the "action" $\mathbf{W}$ takes place in the directions of the row space. $\mathbf{W}$ "doesn't care" about any components of the input in the direction of the null space.

Now consider the transpose matrix $\mathbf{W}^{\mathrm{T}}$, which we get from $\mathbf{W}$ by switching the rows and columns. $\mathbf{W}^{\mathrm{T}}$ is a linear transformation from the output space of $\mathbf{W}$ back into the input space of $\mathbf{W}$. Of course, the row space of $\mathbf{W}^{\mathrm{T}}$ is just the column space of W and vice versa. Given the definitions above, we have that $Im(\mathbf{W})$ is orthogonal to $Null(\mathbf{W}^{\mathrm{T}})$ and $Im(\mathbf{W}^{\mathrm{T}})$ is orthogonal to $Null(\mathbf{W})$ (problem 9.6.2).

ARGUE THAT FOR W TO BE INVERTIBLE, MUST BE A SQUARE MATRIX WITH LINEARLY INDEPENDENT ROWS AND COLUMNS.

---

[2] Mathematically, we say that the input space S is the **direct sum** of $Row(\mathbf{W})$ and $Null(\mathbf{W})$, $S = Row(\mathbf{W})CIRCPLUSNull(\mathbf{W})$.

Problems

Problem 9.6.1 Show that $\mathbf{W}$ is 1-1 is if and only if $Null(\mathbf{W})$ consists of a single vector, namely the 0-vector.

Problem 9.6.2 Show that $Im(\mathbf{W})$ is orthogonal to $Null(\mathbf{W}^{\mathrm{T}})$ and $Im(\mathbf{W}^{\mathrm{T}})$ is orthogonal to $Null(\mathbf{W})$. Use not only the row/column space definitions, but the definitions of $Null$ and Im.

9.7 The Transpose

In the population vector approach, one takes $\mathbf{T} = \mathbf{W}^{\mathrm{T}}$ rather than $\mathbf{T} = \mathbf{W}^{-1}$ as the decoding matrix. Recall that taking the transpose corresponds to each output neuron "voting" for it's preferred direction with a strength that is proportional to it's level of activity. Suppose we apply the transpose of the matrix $\mathbf{U}$ instead of $\mathbf{V}^{-1}$ in the case of finding the optimal linear mapping, *i.e.* we assume

$$\mathbf{T} = \mathbf{U}\mathbf{V}^{\mathrm{T}} \tag{9.12}$$

instead of $\mathbf{T} = \mathbf{U}\mathbf{V}^{-1}$? Going back to the outer product (hidden unit) view of matrix multiplication, we see that

$$\mathbf{T} = \sum_{\alpha} \mathbf{u}^{\alpha}(\mathbf{v}^{\alpha})^{\mathrm{T}} \tag{9.13}$$

This is just the association matrix we get from Hebbian (correlational) learning applied to multiple pairs of inputs and outputs. This transpose approach sets the linear decoding matrix according to the correlation between inputs and outputs. For binary output vectors like spike trains, elements can only be 0 or 1 so that calculating the correlation is equivalent to computing the average input present when there was a spike. Thus, the transpose approach is closely related to the process of calculating the linear receptive field.

We saw in chapter 8 that Hebbian learning resulted in accurate decoding when the input vectors (the output vectors in the decoding case) were decorrelated. To see how this plays out mathematically, we return to the original encoding equation and multiply by the transpose:

$$\begin{aligned} \mathbf{T}\mathbf{V} &= \mathbf{U} & (9.14)\\ \mathbf{T}\mathbf{V}\mathbf{V}^{\mathrm{T}} &= \mathbf{U}\mathbf{V}^{\mathrm{T}} & (9.15) \end{aligned}$$

If the output vectors are decorrelated, then $\mathbf{v}^{\alpha} \cdot \mathbf{v}^{\beta} = 0$ for $\alpha \neq \beta$ and the matrix $\mathbf{V}\mathbf{V}^{\mathrm{T}}$ has 0's except along it's diagonal. If the output vectors are normalized so that $\mathbf{v}^{\alpha} \cdot \mathbf{v}^{\alpha} = 1$, then $\mathbf{V}\mathbf{V}^{\mathrm{T}}$ is equal to the identity matrix and the optimal decoding matrix $\mathbf{T} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$. Usually, the stimulus set used when calculating the receptive is a set of decorrelated inputs. Thus, the optimal mapping from inputs to spikes is given by the average stimulus before a spike. So, to calculate the probability that any given stimulus will produce a spike, one simply takes the inner product of the stimulus with the average stimulus before a spike.

However, if the output vectors are not decorrelated and normalized, then using the transpose will not yield the optimal linear map. What can go wrong? For the most obvious case, consider population coding in the wind detection system of the cricket. (WENT OVER IN CLASS). The bottom line is that for the population vector method to work, the direction vectors must be evenly spread out. For the Hebbian learning case, the corresponding statement is the input patterns to $\mathbf{T}$ (output patterns if $\mathbf{T}$ is a decoding matrix) must be decorrelated (evenly spread out).

9.8 Optimal Maps Revisited

Let's take a step back on the problem of optimal maps and describe things qualitatively in the two basic situations we are considering. In the case of population coding, we are given a pattern of activity and we want to predict what input parameters led to that activity pattern. One problem here is noise. If we simply consider the average (cosine-shaped) tuning curves, there may be no input pattern that would give rise to the exact pattern of activity recorded on any given trial. For example, consider two neurons with exactly opposite preferred direction. On average, the activity of one neuron is modulated up in the exact amount that the other is modulated downward. But on a given trial that exact symmetry is likely to be broken. Geometrically, if there are more neurons than input parameters, then the set of activity patterns driven by the input will be a lower dimensional subspace of the output space ($Im(\mathbf{W})$ in our example above). Adding random noise to each output neuron corresponds to taking a random jump in the output space and hence will likely bump the vector off of the image subspace. The optimal decoding strategy for a given output vector would seem to be to find the closest output pattern that could be obtained without noise, and then use the input vector corresponding to that noise-less output at the predicted input.

If the input or direction vectors aren't evenly spread out, we can find the optimal decoding matrix by "dividing out" these correlations:

$$\mathbf{T} = \mathbf{U}\mathbf{V}^{\mathrm{T}} \left(\mathbf{V}\mathbf{V}^{\mathrm{T}}\right)^{-1} \tag{9.16}$$

But now what is to guarantee that the matrix $(\mathbf{V}\mathbf{V}^{\mathrm{T}})$ is invertible? Well nothing. However, correlation matrices of the form $(\mathbf{V}\mathbf{V}^{\mathrm{T}})$ do have a number of properties that make them much nicer to deal with than matrices in general. First, correlation matrices are square so they have the potential to be invertible. Second, correlation matrices are symmetric in that $(\mathbf{V}\mathbf{V}^{\mathrm{T}})_{ij} = (\mathbf{V}\mathbf{V}^{\mathrm{T}})_{ji}$. The importance of this will be discussed in more detail later in chapter 11. But for right now, it is sufficient to know that $(\mathbf{V}\mathbf{V}^{\mathrm{T}})$ will be invertible as long as the column space of $\mathbf{V}$ spans the domain.

MAIN IDEA: NEED TO DIVIDE OUT CORRELATIONS ON "INPUT" SIDE OF LINEAR MAP. FOR RECEPTIVE FIELDS COMPUTED WITH DECORRELATED STIMULI, THIS ISN'T AN ISSUE. RECEPTIVE FIELD CAN BE USED TO PREDICT SPIKE PROBABILITY FROM A GIVEN STIMULUS. TO PREDICT STIMULUS FROM A SPIKE, NEED TO DIVIDE OUT CORRELATIONS IN THE SPIKE TRAIN. ALSO, CAN USE METHOD IN FORWARD DIRECTION FOR STIMULI THAT AREN'T DECORRELATED, LIKE BIRD SONGS.

Chapter 10

Recurrent Networks

10.1 Recurrent Networks

NOTE MOST OF THIS SECTION IS A REPEAT.

So far, we have only considered “feedforward” networks, where the activity in the input layer determined the activity in the next layer and so on. We now introduce “feedback” or “recurrent” networks. As opposed to feedforward networks, which only had connections between layers, recurrent networks have synaptic connections between neurons in the same layer. The standard example that we will consider has two layers of neurons with recurrent connections in the output layer (see figure 10.1). This network has two different weight matrices, the matrix **W** of feedforward weights and the matrix **T** of recurrent weights. These networks are known as feedback networks because the recurrent connections allow the output of a neuron to influence it’s input, either directly or indirectly via its effect on other neurons.

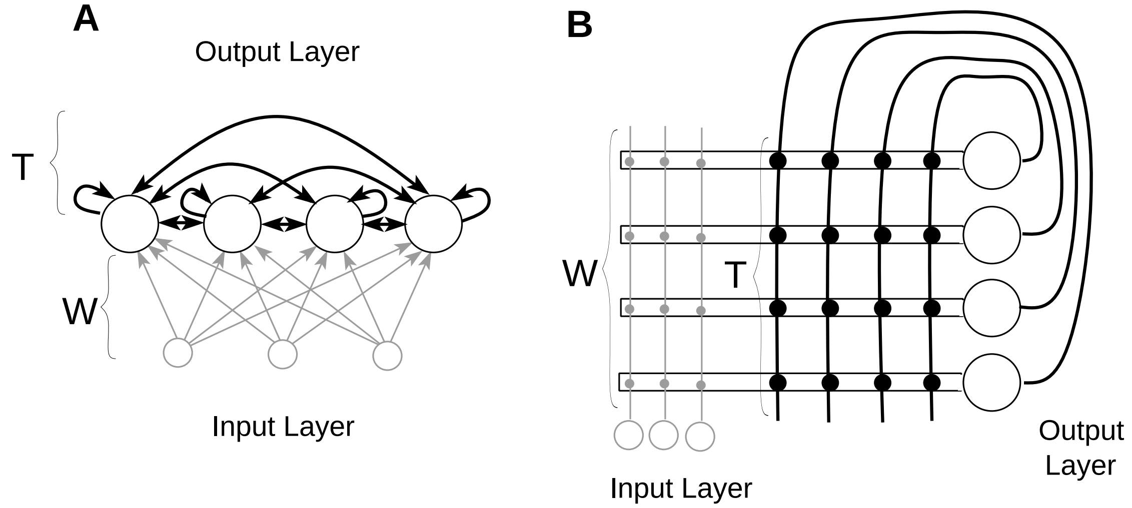


Figure 10.1: Two representations of a two-layer, recurrent network.

Let’s consider a recurrent network of linear neurons, and focus on neurons in the output layer:

$$r_i = \sum_j W_{ij} q_i + \sum_k T_{ik} r_k \tag{10.1}$$

Note that given an input pattern, we can’t directly compute the outputs because the output variables r_i fall on both sides of the equation. This is just a mathematical restatement of the fact that the neurons are reciprocally connected.

The easiest way to proceed is to rewrite the equation using vector notation:

$$\mathbf{r} = \mathbf{W}\mathbf{q} + \mathbf{T}\mathbf{r} \tag{10.2}$$

To get further we'll need to introduce a useful and important matrix:

Definition 14 *The identity matrix* $\mathbf{I}$ *is the matrix such that for all vectors* $\mathbf{v}$, $\mathbf{I}\mathbf{v} = \mathbf{v}$.

$\mathbf{I}$ acts like the number 1 in ordinary multiplication, and is a square matrix with 1's along it's diagonal and zeros elsewhere (problem **??**). Now we can define the inverse of a matrix $\mathbf{W}$ (if it exists) to be the matrix $\mathbf{W}^{-1}$ such that $\mathbf{W}^{-1}\mathbf{W} = \mathbf{I}$. Returning to the recurrent network equations we can write,

$$\begin{aligned} \mathbf{r} &= \mathbf{I}\mathbf{r} = \mathbf{W}\mathbf{q} + \mathbf{T}\mathbf{r} && (10.3) \\ (\mathbf{I} - \mathbf{T})\mathbf{r} &= \mathbf{W}\mathbf{q} && (10.4) \\ \mathbf{r} &= (\mathbf{I} - \mathbf{T})^{-1}\mathbf{W}\mathbf{q} && (10.5) \end{aligned}$$

Therefore, the net effect of the recurrent connections is to perform a linear mapping, $\widetilde{\mathbf{T}} = (\mathbf{I} - \mathbf{T})^{-1}$, on the pattern $\mathbf{W}\mathbf{q}$ of total synaptic input arriving at the output layer. In this chapter (and the next few) we will focus on how the matrix $\widetilde{\mathbf{T}}$ transforms the total feedforward input $\mathbf{W}\mathbf{q}$. For simplicity we will replace $\mathbf{W}\mathbf{q}$ with $\mathbf{q}$, and interpret it as the pattern of the total input arriving from the input layer.

> Biological Aside. In trying to understand information processing within hierarchical networks, it is often useful to conceptually divide inputs into feedforward or bottom-up inputs coming from the previous processing layer, lateral or recurrent inputs from other neurons at the same layer, and top-down inputs coming from the next layer up the processing hierarchy. Note that the term feedback connection can be used for either lateral or top-down connections. Inputs that cannot be easily put within this hierarchical framework are often thought of as "modulatory" inputs. Parsing the role of the different inputs in the visual cortex has been the subject of a number of experiments and models.

For "well-behaved" (to be defined in chapter 12) recurrent matrices $\mathbf{T}$,

$$(\mathbf{I} - \mathbf{T})^{-1} = \mathbf{I} + \mathbf{T} + \mathbf{T}^2 + \mathbf{T}^3 + \dots \tag{10.6}$$

We define $\mathbf{T}^2$ to be the matrix $\mathbf{T}\mathbf{T}$, $\mathbf{T}^3 = \mathbf{T}\mathbf{T}\mathbf{T}$, etc. Equation (10.6) has a natural biological interpretation. The activity in the output layer results from the direct input, $\mathbf{q} = \mathbf{I}\mathbf{q}$, plus the input resulting from $\mathbf{q}$ being passed through the recurrent weights, $\mathbf{T}\mathbf{q}$, plus the input resulting from this activity being passed through the recurrent weights again, $\mathbf{T}\mathbf{T}\mathbf{q}$, etc. In other words, $\widetilde{\mathbf{T}}_{ik}$ *describes the degree to which the feedforward input arriving at the kth neuron drives the output of the ith neuron, by summing up over all possible pathways of information flow.*

Problems

Problem 10.1.1 Show that the identity matrix for N dimensional vectors is an $N \times N$ (square) matrix, with 1's along it's diagonal and zeros elsewhere. *Hint: show that for every way in which the assumptions would not hold, one could produce a vector* $\mathbf{v}$ *with* $\mathbf{I}\mathbf{v} \neq \mathbf{v}$.

Problem 10.1.2 Justify the expansion in equation (10.6) under the assumption that $\lim_{n\to\infty} \mathbf{T}^n = 0$.

10.2 An Example

We will present the main techniques related to linear recurrent networks in the context of a simple example containing only two output units (see figure **??**). Using two outputs will allow us to both handle numerical examples and to visualize the corresponding geometry. However, coming up with meaningful biological examples is a bit trickier. The easiest interpretation for now is that the two output units represent two populations of neurons representing two different features or concepts in the world.

According to equation (10.5), we are really interested in the matrix $\widetilde{\mathbf{T}} = (\mathbf{I} - \mathbf{T})^{-1}$. For 2×2 matrices, if

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{10.7}$$

then

$$\mathbf{M}^{-1} = \frac{1}{\mathrm{Det}(\mathbf{M})} \begin{bmatrix} M_{22} & -M_{12} \\ -M_{21} & M_{11} \end{bmatrix} \tag{10.8}$$

where $\mathrm{Det}(\mathbf{M}) = M_{11}M_{22} - M_{12}M_{21}$ is the **determinant** of the matrix (see problem 10.2.1). We will discuss the determinant in more detail below.

Definition 15 *A matrix* $\mathbf{M}$ *is a* **symmetric matrix** *if* $M_{ij} = M_{ji}$ *for all* i *and* j*.*

Symmetric matrices have a number of very nice mathematical properties and can be found in many network models. We will discuss these properties below. While the assumption of symmetric connections is not supported at the level of single neurons, it is often a reasonable assumption at the level of neural populations. Note that if the matrix $\mathbf{M}$ is symmetric, the $\mathbf{M}^{-1}$ is symmetric as well.

Using this formula we find that

$$\begin{aligned}
\widetilde{\mathbf{T}}_{11} &= \frac{1 - T_{22}}{(1 - T_{11})(1 - T_{22}) - T_{12}T_{21}} && (10.9) \\
\widetilde{\mathbf{T}}_{12} &= \frac{T_{12}}{(1 - T_{11})(1 - T_{22}) - T_{12}T_{21}} && (10.10) \\
\widetilde{\mathbf{T}}_{21} &= \frac{T_{21}}{(1 - T_{11})(1 - T_{22}) - T_{12}T_{21}} && (10.11) \\
\widetilde{\mathbf{T}}_{22} &= \frac{1 - T_{11}}{(1 - T_{11})(1 - T_{22}) - T_{12}T_{21}} && (10.12) \\
& && (10.13)
\end{aligned}$$

Problems

Problem 10.2.1 (E) Confirm the validity of equation (10.8).

10.2.1 Non-interaction Case

The network becomes significantly simpler if we remove the connections between the two outputs, *i.e.* $T_{12} = T_{21} = 0$. Using the equations above,

$$\widetilde{\mathbf{T}} = \begin{bmatrix} \frac{1}{1-T_{11}} & 0 \\ 0 & \frac{1}{1-T_{22}} \end{bmatrix} \tag{10.14}$$

Since the only non-zero terms in the matrix lie along the diagonal, we say that $\widetilde{\mathbf{T}}$ is a **diagonal matrix**. The solution to the equation $\mathbf{r} = \widetilde{\mathbf{T}}\mathbf{q}$ is given by

$$\begin{aligned} r_1 &= \widetilde{\mathbf{T}}_{11} q_1 = q_1/(1 - T_{11}) && (10.15) \\ r_2 &= \widetilde{\mathbf{T}}_{22} q_2 = q_2/(1 - T_{22}) && (10.16) \end{aligned}$$

The network simply multiplies input i by a gain factor of $1/(1 - T_{ii})$. If $T_{ii} = 0$, then the gain is 1 and the network simply replicates the input ($\widetilde{\mathbf{T}} = \mathbf{I}$). As T_{ii} grows larger and approaches 1, then the gain $1/(1 - T_{ii})$ approaches infinity. At this point the positive feedback from self-excitation makes the system unstable. We will explore the issue of stability in chapter 12. For negative values of T_{ii}, the recurrent connections implement negative feedback and the gain is less than 1.

10.3 Coordinates and Bases

In many two dimensional problems, it can be useful to change coordinates to "sum and difference coordinates."

> **Example 10.3.1** Suppose we perform the following hypothetical experiment, aimed at determining the "binocularity" of visual cortical neurons. We first present a stimulus to the right eye, counting the number of spikes produced by that neuron in response. We then present the same stimulus to the left eye and note the response. For each neuron, the outcome of the experiment can be described by a two dimensional vector $\mathbf{r} = [\mathbf{r}^{right}, \mathbf{r}^{left}]^{\mathrm{T}}$. But the same information could be represented using a different set of coordinates: $\mathbf{r} = [\mathbf{r}^{sum}, \mathbf{r}^{diff}]^{\mathrm{T}}$, where $\mathbf{r}^{sum} = \mathbf{r}^{right} + \mathbf{r}^{left}$ and $\mathbf{r}^{diff} = \mathbf{r}^{right} - \mathbf{r}^{left}$. The difference coordinate is one measure of binocularity of the cell, *i.e.* how much the cell responds more to one eye than the other. The sum coordinate captures the total responsiveness of the neuron to the stimuli presented.

Let's follow this sum and difference approach and consider an input vector $\mathbf{q} = [2, 6]^{\mathrm{T}}$. We can use vector notation to write the sum $q_1 + q_2 = 8$ as $[1, 1]^{\mathrm{T}}\mathbf{q}$, and the difference $q_1 - q_2 = -4$ as $[1, -1]^{\mathrm{T}}\mathbf{q}$. In the sum and difference coordinates, $\mathbf{q} = [8, -4]^{\mathrm{T}}$. We'd like to continue to think of activity patterns as vectors, but now we've represented the same pattern of input $\mathbf{q}$ using two sets of numbers: $\mathbf{q} = [2, 6]^{\mathrm{T}}$ and $\mathbf{q} = [8, -4]^{\mathrm{T}}$. To keep thing clear, we will use the notation $\mathbf{q} = [2, 6]^{\mathrm{T}}_{\mathbf{S}}$ and $\mathbf{q} = [8, -4]^{\mathrm{T}}_{\mathbf{D}}$ (**S** for the "standard" coordinates, **D** for sum and *difference* coordinates).

In what sense can these two lists of numbers represent the same vector? The key to keeping things straight is to make a distinction between vectors themselves and their expressions as lists of coordinates. To sort this out, let's consider our fundamental objects to be patterns of input, taken somewhat abstractly. Surely we can add patterns of input and scale patterns of input no matter what coordinates we use to describe them. What we need to clarify is the relationship between the pattern of input $\mathbf{q}$ and the coordinates $[2, 6]^{\mathrm{T}}_{\mathbf{S}}$ and $[8, -4]^{\mathrm{T}}_{\mathbf{D}}$. Consider the input patterns $\mathbf{e}_i$ where input i has magnitude 1 and the other input is 0. Then the input vector $\mathbf{q} = 2\mathbf{e}^1 + 6\mathbf{e}^2$. Now let $\mathbf{d}^1$ be the input pattern where both inputs are at 1/2, and let $\mathbf{d}^2$ be the input pattern where input 1 is equal to 1/2 and input 2 is equal to -1/2. Then $\mathbf{q} = 8\mathbf{d}^1 - 4\mathbf{d}^2$.

Definition 16 *A* **basis B** *for a vector space V is a set of vectors $\mathbf{B} = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n\}$ in V such that any vector $\mathbf{u}$ in V can be written as a linear combination of vectors in $\mathbf{B}$. The vectors in B are said to* **span** *V. If we write $\mathbf{u} = \sum_i c_i \mathbf{v}^i$, the list of numbers $[c_1, c_2, \ldots, c_n]^{\mathrm{T}}_{\mathbf{B}}$ are the* **coordinates** *of $\mathbf{u}$ in the basis $\mathbf{B}$. If the vectors in $\mathbf{B}$ are mutually orthogonal ($\mathbf{v}^i \cdot \mathbf{v}^j = 0$ for $i \neq j$) and normalized ($\mathbf{v}^i \cdot \mathbf{v}^i = 1$), then the basis $\mathbf{B}$ is said to be* **orthonormal**.

$\{\mathbf{e}^1, \mathbf{e}^2\}$ is the **standard basis** in two dimensions, and $\{\mathbf{d}^1, \mathbf{d}^2\}$ is the appropriate basis for the sum and difference coordinates. Note that we can write $\mathbf{d}^1 = [1/2, 1/2]^T_{\mathbf{S}}$ and $\mathbf{d}^1 = [1/2, -1/2]^T_{\mathbf{S}}$. Generally, when we write a vector as a list of numbers, there usually isn't any special notation. Most commonly, it is implicitly assumed that coordinates relate to the standard basis. In other cases, the basis (and hence the meaning of the coordinates) is generally clear.

Orthonormal bases have a number of nice properties. In particular, if $\mathbf{B} = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n\}$ is an orthonormal basis then $\mathbf{u} = \sum_i (\mathbf{u} \cdot \mathbf{v}^i)\mathbf{v}^i$ for any vector $\mathbf{u}$ (problem 10.3.2). Since the basis vectors are normalized, coordinate i is simply the length of the projection of $\mathbf{u}$ onto the basis vector $\mathbf{v}^i$. Geometrically, the vector $\mathbf{e}^1$ lies along the x-axis and $\mathbf{e}^2$ lies along the y-axis. Since the standard basis is orthonormal, associating coordinates and vectors follows the cartesian procedure shown in figure 6.4. The sum and difference basis is not orthonormal, however, since the vectors $\mathbf{d}^1$ and $\mathbf{d}^2$ are not normalized to length 1.

The details of *changing* coordinates can get a little confusing. For that reason, I've separated it into sections 10.7.1 and 10.7.2 below. You can get by with the material I presented here.

Problems

Problem 10.3.1 (E) Write the standard basis vectors in sum and difference coordinates

Problem 10.3.2 Show that if $\mathbf{B} = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n\}$ is an orthonormal basis then $\mathbf{u} = \sum_i (\mathbf{u} \cdot \mathbf{v}^i)\mathbf{v}^i$ for any vector $\mathbf{u}$.

Problem 10.3.3 Show that for any basis $\mathbf{B} = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n\}$, $\mathbf{v}^1 = [1, 0, \ldots, 0]^T_{\mathbf{B}}$.

Now that we've clarified the relationship between vectors in general and lists of numbers, we need to clarify the relationship between matrices (two dimensional arrays of numbers) and the linear transformations that they implement. To keep things simple let's consider a linear transformation $\mathbf{M}$ that takes vectors in two-dimensional space and transforms them into other vectors in two-dimensional space. Suppose we start with the standard basis $\{\mathbf{e}^1, \mathbf{e}^2\}$, and use our procedure for matrix multiplication.

$$\mathbf{M}\mathbf{e}^1 = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} M_{11} \\ M_{12} \end{bmatrix} = M_{11}\mathbf{e}^1 + M_{21}\mathbf{e}^2 \tag{10.17}$$

$$\mathbf{M}\mathbf{e}^2 = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} M_{21} \\ M_{22} \end{bmatrix} = M_{12}\mathbf{e}^1 + M_{22}\mathbf{e}^2 \tag{10.18}$$

So, given a linear transformation $\mathbf{M}$ and a basis $\mathbf{B} = \{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n\}$, we can represent $\mathbf{M}$ as an array of numbers such that $\mathbf{M}_{ij}$ is the ith coordinate of the vector $\mathbf{v}^j$ in the basis $\mathbf{B}$. That is, $\mathbf{M}\mathbf{v}^j = \sum_i \mathbf{M}_{ij}\mathbf{v}_i$.

10.4 Eigenvectors

For an example of expressing a matrix in new coordinates, let's return to our simple recurrent network example, and suppose that each output unit has the same pattern of connectivity. Therefore, the matrices $\mathbf{T}$ and $\widetilde{\mathbf{T}}$ are symmetric. Moreover, $\widetilde{\mathbf{T}}_{11} = \widetilde{\mathbf{T}}_{22}$. To be concrete let's consider the matrix

$$\widetilde{\mathbf{T}} = \begin{bmatrix} 1.5 & -1 \\ -1 & 1.5 \end{bmatrix}_{\mathbf{S}} \tag{10.19}$$

where we have used the subscript $\mathbf{S}$ to specify that the transformation $\widetilde{\mathbf{T}}$ is written using the standard basis.

Now let's see what happens if we apply the matrix $\widetilde{\mathbf{T}}$ to the sum and difference basis vectors $\mathbf{d}^1 = [1/2, -1/2]^{\mathrm{T}}_{\mathbf{S}}$ and $\mathbf{d}^2 = [1/2, -1/2]^{\mathrm{T}}_{\mathbf{S}}$.

$$\begin{aligned} \widetilde{\mathbf{T}}\mathbf{d}^1 &= \widetilde{\mathbf{T}}[1/2, 1/2]^{\mathrm{T}}_{\mathbf{S}} = [.75 - .5, -.5 + .75]^{\mathrm{T}}_{\mathbf{S}} = .5[1/2, -1/2]^{\mathrm{T}}_{\mathbf{S}} = .5\mathbf{d}^1 + 0\mathbf{d}^2 && (10.20) \\ \widetilde{\mathbf{T}}\mathbf{d}^2 &= \widetilde{\mathbf{T}}[1/2, -1/2]^{\mathrm{T}}_{\mathbf{S}} = [.75 + .5, -.5 - .75]^{\mathrm{T}}_{\mathbf{S}} = 2.5[1/2, -1/2]^{\mathrm{T}}_{\mathbf{S}} = 0\mathbf{d}^1 + 2.5\mathbf{d}^2 && (10.21) \end{aligned}$$

Thus, when expressed in sum and difference coordinates, the transformation $\widetilde{\mathbf{T}}$ becomes

$$\widetilde{\mathbf{T}} = \begin{bmatrix} .5 & 0 \\ 0 & 2.5 \end{bmatrix}_{\mathbf{D}} \tag{10.22}$$

We say that the basis $\mathbf{D}$ **diagonalizes** the transformation $\widetilde{\mathbf{T}}$.

In diagonalizing $\widetilde{\mathbf{T}}$, sum and difference coordinates transform the problem back into a non-interacting case. However, the lack of interaction is not between individual output units but between separate *patterns* of activity. Thus, $^{\mathrm{T}}$ has the effect of compressing the sum of the inputs by a factor of two, no matter what the difference between input 1 and 2, while scaling the difference between the inputs by a factor of 2.5, no matter what the sum of the inputs might be. Geometrically, the vectors $\mathbf{d}^1$ and $\mathbf{d}^2$ have the very special property that multiplying by the matrix $\widetilde{\mathbf{T}}$ simply scales these vectors without changing their direction. Such vectors are called **eigenvectors** and the values .5 and 2.5 that represent how much these vectors are scaled are the corresponding **eigenvalues**. More formally,

Definition 17 *A vector* $\mathbf{v}$ *is called an* **eigenvector** *for a linear transformation* $\mathbf{M}$ *if* $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$. *The constant* λ *is called the* **eigenvalue** *for the eigenvector* $\mathbf{v}$.

It is important to point out, that the definition of an eigenvector and eigenvalue is "coordinate free", *i.e.* the eigenvectors of a transformation will be the same vector no matter what coordinates they are expressed in and the corresponding eigenvalue will always be the same. If one can find a basis consisting of eigenvectors, this basis is called an **eigenbasis**. Thus, the basis $\{\mathbf{d}^1, \mathbf{d}^2\}$ is an eigenbasis for the transformation $\widetilde{\mathbf{T}}$.

This example illustrates a fairly general strategy for solving a number of problems in linear algebra. If one can find an eigenbasis for a linear transformation then one can view the transformation as a series of independent scalings in a number of directions. Luckily, *any symmetric matrix* $\mathbf{M}$ *has an eigenbasis. Moreover, one can find an orthonormal eigenbasis for* $\mathbf{M}$. This property makes symmetric matrices particulary amenable to analysis, and that makes them particularly common in the field of computational neuroscience. However, it is important to remember that not all matrices are symmetric, nor do all of them have an eigenbasis.

In our example, we didn't use any procedure to find the eigenvectors, we just applied the matrix to the sum and difference basis vectors and confirmed that indeed these were eigenvectors. While this seems like cheating, guessing the answer and then proving it works is a tried and true methodology in applied mathematics. On further reflection, the sum and difference coordinates weren't such a far out guess. This is because they are aligned with the symmetries of the problem. In particular, the connectivity of each output unit is identical. Therefore, the solution to the problem shouldn't be different if we switch the indices 1 and 2. Geometrically this means that the eigenvectors should be the same if we exchange the x and y axes. The sum and difference eigenvectors lie in the two directions that aren't changed by such a permutation. Although this method of exploiting the symmetries of the problem is successful in many cases, it isn't always so.

There is a more general method for finding eigenvectors and eigenvalues that is presented below in section 10.7.3 for the interested reader. More common nowadays is to rely on software packages to solve for the eigenvectors and eigenvalues.

Problems

Problem 10.4.1 Find the matrix $\mathbf{T}$ that gives the specific $\widetilde{\mathbf{T}}$ in equation (10.19).

Problem 10.4.2 Write out the first few terms in the expansion $\widetilde{\mathbf{T}} = (\mathbf{I} - \mathbf{T})^{-1} = I + \mathbf{T} + \mathbf{T}^2 + \mathbf{T}^3 + \ldots$. Use the specific values for $\mathbf{T}$ found in problem 10.4.1.

Problem 10.4.3 Find the conditions on a 2×2 symmetric matrix

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{10.23}$$

that ensures that the sum and difference basis is an eigenbasis for $\mathbf{M}$.

Problem 10.4.4 Another set of coordinates that is often convenient in two dimensional problems are the average and deviation coordinates:

$$(q_1, q_2) \rightarrow (q_{avg}, q_{dev}) = \left(\frac{q_1 + q_2}{2}, \frac{q_1 - q_2}{2} \right)$$

Find the basis vectors for this coordinate system.

Problem 10.4.5 Show that any matrix is diagonal when expressed in its eigenbasis, and the eigenvalues are the entries along the diagonal.

Problem 10.4.6 Find an orthonormal eigenbasis for the matrix $\widetilde{\mathbf{T}}$ in equation (10.19).

In our example, the effect of the recurrent network was to expand difference between the inputs by a factor of 2.5. Such an expansion of the difference is expected since the connections between the output units are negative. Thus, each unit tends to inhibit the other one, increasing the difference in activity levels. However, if we take

$$\widetilde{\mathbf{T}} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \tag{10.24}$$

Once again the sum and difference coordinates form an eigenbasis, with the sum direction eigenvalue equal to 4, and the difference eigenvalue equal to 2. So the network still expands the difference between the values even though there is a positive connection between the two output units. Why is this so? The easiest way to clarify the issue is to go back to a non-interacting network ($\mathbf{T}_{12} = \mathbf{T}_{21} = 0$) and take $\mathbf{T}_{11} = \mathbf{T}_{22} = 1/2$. Then

$$\widetilde{\mathbf{T}} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = 2\mathbf{I} \tag{10.25}$$

Therefore, $\widetilde{\mathbf{T}}$ simply scales the input vector by a factor of two without changing the relative magnitude of the two components. But of course this uniform expansion will increase the difference between the components as well as increasing their sum. What is needed for the network as implementing a true competition between the inputs is for the difference to increase more than the sum (or for the difference to decrease less than the sum). One can show that this happens exactly when the connections between the units are negative (problem 10.5.1).

10.5 Interpretations

Having an eigenbasis for a linear transformation makes it quite easy to get a geometric picture of the transformation. In particular, the eigenvectors determine the directions in space along which vectors are "stretched" will the eigenvalues giving the magnitude of the stretch. Negative eigenvalues lead to a "flip" along with a stretch.

FIGURE

It is hard to overemphasize the importance of eigenvectors for computational neuroscience. For any system that is linear or nearly linear, the eigenvectors are a guide to breaking a problem into it's component parts. While it is not absolutely necessary to understand the mechanics of finding the eigenvectors (this is presented below), understanding the conceptual importance of eigenvectors is key to understanding a number of papers in the literature.

Problems

Problem 10.5.1 Show that the whether the difference component increases more (or decreases less) than the sum component depends on the sign of the connection between the two output units. (Assume $\mathbf{T}_{11} = \mathbf{T}_{22}$.)

10.6 Coordinate Free Quantities

Although we relegate the details of some of the linear algebra to the section below, there are a two important quantities that one can extract from square matrices that are invariant under a change of coordinates. We've already seen such quantities, namely the eigenvectors and eigenvalues of linear transformation are the same no matter what coordinates are used to express them. Since the eigenvalues, let's call them $\lambda_1, \lambda_2, \lambda_3, \ldots$, don't change under a change of coordinates, it is not surprising that their sum and product $\sum_i \lambda_i$ and $\prod_i \lambda_i$ are also coordinate free (assuming that we can find an eigenbasis). What is surprising is that these numbers can be extracted in a relatively straightforward way from the entries in the matrix describing a linear transformation, and these *do* change as one changes coordinates.

For example, one can define the **trace** of a matrix $\mathbf{M}$ as the sum of the diagonal elements, $\text{Trace}(\mathbf{M}) = \sum_i \mathbf{M}_{ii}$. If one expresses a linear transformation using the eigenbasis, then the corresponding matrix is diagonal, and the trace is equal to the sum of the eigenvalues. But the trace is coordinate free. That means the *sum* of the diagonal elements of a matrix does not change as one changes coordinates, even though the individual entries may indeed change. In our original example above, $\text{Trace}(\widetilde{\mathbf{T}})$ can be computed from matrix in the original coordinates (equation 10.19) or after it had been diagonalized (equation 10.22). Thus one can get some information about the eigenvalues of a matrix, even before finding the eigenvectors.

A second quantity that can be extracted is the **determinant**. The formula for the determinant for a general square matrix is a bit complicated. For a 2 × 2 matrix it is equal to $M_{11}M_{22} - M_{12}M_{21}$. For matrices with an eigenbasis, the determinant is equal to the product of the eigenvalues. Therefore, the determinant is coordinate free and applying the formula to the entries of two different matrices will come up with the same answer if the two matrices express the same underlying linear transformation in different coordinates. A geometric way to interpret the determinant is as the expansion/compression ratio for volumes under the linear transformation, *i.e.* it is the volume of the image of a cube with area 1. If the determinant is negative, then it tells you that the transformation has a net flip.

10.7 The Linear Algebra of Coordinates and Eigenvectors

In this section we will expand on the above ideas with a bit more rigor.

SECTION NEEDS REWORKING/COMPLETION.

10.7.1 Changing Coordinates - Vectors

The goal of this section is to determine how to take a vector expressed as a list of coordinates in one basis, the "old basis," $\mathbf{Q} = \{\mathbf{q}^1, \mathbf{q}^2, \ldots, \mathbf{q}^N\}$, and express it as a list of coordinates in another basis, the "new basis," $\mathbf{R} = \{\mathbf{r}^1, \mathbf{r}^2, \ldots, \mathbf{r}^N\}$. In other words, suppose we are given a vector $\mathbf{v} = [v_1^{\mathbf{Q}}, v_2^{\mathbf{Q}}, \ldots, \mathbf{v}_N^{\mathbf{Q}}]^{\mathrm{T}}$, how do we determine the coordinates $v_i^{\mathbf{R}}$ so that $[v_1^{\mathbf{R}}, v_2^{\mathbf{R}}, \ldots, \mathbf{v}_N^{\mathbf{R}}]^{\mathrm{T}}$ describes the *same* vector $\mathbf{v}$ relative to the new basis $\mathbf{R}$. To do this we must assume that we know how to express the new basis vectors $\{\mathbf{r}^1, \mathbf{r}^2, \ldots, \mathbf{r}^N\}$ as lists of coordinates in the old basis $\mathbf{Q}$. The task will be to use that information to be able to change coordinates, *i.e.* to take *any* vector that is given as a list of old coordinates, and transform that list so that the vector is expressed as a list of new coordinates. Note that expressing the new basis vectors as coordinates in the new basis is trivial:

$$\begin{aligned}
\mathbf{r}^1 &= 1\mathbf{r}^1 + 0r^2 + \ldots + 0\mathbf{r}^N = [1, 0, \ldots, 0]_{\mathbf{R}}^{\mathrm{T}} \\
\mathbf{r}^2 &= 0\mathbf{r}^1 + 1r^2 + \ldots + 0\mathbf{r}^N = [0, 1, \ldots, 0]_{\mathbf{R}}^{\mathrm{T}} \\
&\vdots \\
\mathbf{r}^N &= 0\mathbf{r}^1 + 0r^2 + \ldots + 1\mathbf{r}^N = [0, 0, \ldots, 1]_{\mathbf{R}}^{\mathrm{T}}
\end{aligned}$$

The transformation that takes in lists of numbers representing vectors in the old coordinates and transforms them into lists of numbers in the new coordinates is a linear transformation. Therefore, changing coordinates can be accomplished by matrix multiplication. We will denote the matrix that transforms vectors in the basis $\mathbf{Q}$ to the basis $\mathbf{R}$ as ${}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}}$, *i.e.* $[v_1^{\mathbf{R}}, v_2^{\mathbf{R}}, \ldots, \mathbf{v}_N^{\mathbf{R}}]^{\mathrm{T}} = {}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}}[v_1^{\mathbf{Q}}, v_2^{\mathbf{Q}}, \ldots, \mathbf{v}_N^{\mathbf{Q}}]^{\mathrm{T}}$.

Note that solving the reverse problem is easy: if we are given a vector $\mathbf{v} = [v_1^{\mathbf{R}}, v_2^{\mathbf{R}}, \ldots, \mathbf{v}_N^{\mathbf{R}}]^{\mathrm{T}}$ expressed in the basis $\mathbf{R}$, it is easy to express $\mathbf{v}$ in the basis $\mathbf{Q}$. By definition, we have the following vector equation:

$$\mathbf{v} = v_1^{\mathbf{R}}\mathbf{r}^1 + v_2^{\mathbf{R}}\mathbf{r}^2 + \ldots + v_N^{\mathbf{R}}\mathbf{r}^N \tag{10.26}$$

To get the coordinates of $\mathbf{v}$ in the basis $\mathbf{Q}$ we simply add component by component:

$$\mathbf{v} = [\sum_k v_k^{\mathbf{R}} r_1^k, \sum_k v_k^{\mathbf{R}} r_2^k, \ldots, \sum_k v_k^{\mathbf{R}} r_N^k]^{\mathrm{T}} \tag{10.27}$$

These equations can also be expressed in matrix notation, as $[v_1^{\mathbf{Q}}, v_2^{\mathbf{Q}}, \ldots, \mathbf{v}_N^{\mathbf{Q}}]^{\mathrm{T}} = {}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}[v_1^{\mathbf{R}}, v_2^{\mathbf{R}}, \ldots, \mathbf{v}_N^{\mathbf{R}}]^{\mathrm{T}}$, where

$${}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{r}^1 & \mathbf{r}^2 & \ldots & \mathbf{r}^N \\ \vdots & \vdots & & \vdots \end{bmatrix} \tag{10.28}$$

i.e. the columns of ${}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}$ are the coordinates of the basis vectors $\{\mathbf{r}^1, \mathbf{r}^2, \ldots, \mathbf{r}^N\}$ expressed in the basis $\mathbf{Q}$.

We now return to the problem of finding the matrix ${}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}}$. If we start with a vector $\mathbf{v}$ in the old coordinates, transform it to the new coordinates by multiplying by ${}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}}$, and transform it *back* to the old coordinates by multiplying by ${}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}$, we should get back to same vector. Mathematically, ${}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}\,{}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}}[v_1^{\mathbf{Q}}, v_2^{\mathbf{Q}}, \ldots, \mathbf{v}_N^{\mathbf{Q}}]^{\mathrm{T}} = [v_1^{\mathbf{Q}}, v_2^{\mathbf{Q}}, \ldots, \mathbf{v}_N^{\mathbf{Q}}]^{\mathrm{T}}$. In other words, ${}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}} = {}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{-1}$. As before when we discussed the optimal population decoding, we will not describe how to actually compute the inverse. We do note, however, that ${}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{-1} = ({}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{\mathrm{T}}\,{}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}})^{-1}\,{}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{\mathrm{T}}$, and that $({}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{\mathrm{T}}\,{}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}})$ is the matrix of correlations among the basis vectors $\mathbf{r}^k$. When the basis vectors form an orthonormal basis, $({}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{\mathrm{T}}\,{}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}) = \mathbf{I}$ and so ${}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}} = {}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{-1} = {}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{\mathrm{T}}$, *i.e.*

$${}_{\mathbf{R}}\mathbf{C}_{\mathbf{Q}} = \begin{bmatrix} \ldots & \mathbf{r}^1 & \ldots \\ \ldots & \mathbf{r}^2 & \ldots \\ & \vdots & \\ \ldots & \mathbf{r}^N & \ldots \end{bmatrix} \tag{10.29}$$

Given the definition of matrix multiplication, equation (10.29) says that the kth coordinate of the vector $\mathbf{v}$ in the basis $\mathbf{R}$ is equal to $\mathbf{r}^k \cdot \mathbf{v}$ (see problem **??**). Equation (10.29) can also be interpreted geometrically: *the kth coordinate of a vector* $\mathbf{v}$ *in the orthonormal basis* $\mathbf{R} = \{\mathbf{r}^1, \mathbf{r}^2, \ldots, \mathbf{r}^N\}$ *can be found by projecting* $\mathbf{v}$ *onto* $\mathbf{r}^k$. When the vectors $\mathbf{r}^k$ are not orthonormal, the change of coordinates is accomplished by projection onto the basis vectors $\mathbf{r}^k$, *followed by a compensation for the correlations among the basis vectors via multiplication by* $({}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}}^{\mathrm{T}}\,{}_{\mathbf{Q}}\mathbf{C}_{\mathbf{R}})^{\mathrm{T}}$.

10.7.2 Changing Coordinates - Matrices

Now we have clarified the relationship between a vector and a list of numbers that describes that vector, *i.e.* we know how to express vectors in any given coordinate system or basis. We also know how to change coordinates so as to express this same vector using a different set of numbers. We have also introduced matrix multiplication as a way of performing linear transformations from one vector space to another. Certainly, when we choose to express vectors in different coordinates, the elements of the matrix defining a given linear transformation must also change. Again, we have disconnected the notion of an abstract object – in this case a linear transformation – and the numbers used to express that object.

Suppose we have a linear transformation $\mathbf{W} : \Re^N \rightarrow \Re^P$, and we have a basis $\mathbf{Q} = \{\mathbf{q}^1, \ldots, \mathbf{q}^N\}$ for $\Re^N$ and a basis $\mathbf{R} = \{\mathbf{r}^1, \ldots, \mathbf{r}^P\}$ for $\Re^P$. Let ${}_{\mathbf{R}}\mathbf{W}_{\mathbf{Q}}$ be the matrix of numbers that represents $\mathbf{W}$ using coordinates obtained from expressing vectors in these bases. Now suppose we express vectors in $\Re^N$ in a new basis $\tilde{\mathbf{Q}}$ and vectors in $\Re^P$ in a new basis $\tilde{\mathbf{R}}$. How do we find ${}_{\tilde{\mathbf{R}}}\mathbf{W}_{\tilde{\mathbf{Q}}}$? Suppose we are given $\mathbf{v}^{\tilde{\mathbf{Q}}}$, a vector in $\Re^N$ expressed in the new basis $\tilde{\mathbf{Q}}$. To calculate ${}_{\tilde{\mathbf{R}}}\mathbf{W}_{\tilde{\mathbf{Q}}}\mathbf{v}^{\tilde{\mathbf{Q}}}$ we will simply take a round-about path. First we translate $\mathbf{v}^{\tilde{\mathbf{Q}}}$ back into the original coordinates by applying the matrix ${}_{\mathbf{Q}}\mathbf{C}_{\tilde{\mathbf{Q}}}$ as calculated in the previous section, *i.e.* $\mathbf{v}^{\mathbf{Q}} ={}_{\mathbf{Q}} \mathbf{C}_{\tilde{\mathbf{Q}}}\mathbf{v}^{\tilde{\mathbf{Q}}}$. Now we use the matrix of numbers ${}_{\mathbf{R}}\mathbf{W}_{\mathbf{Q}}$ representing the transformation $\mathbf{W}$ in the original bases to transform $\mathbf{v}^{\mathbf{Q}}$ into a vector ${}_{\mathbf{R}}\mathbf{W}_{\mathbf{Q}}\mathbf{v}^{\mathbf{Q}} ={}_{\mathbf{R}} \mathbf{W}_{\mathbf{Q}}\,{}_{\mathbf{Q}}\mathbf{C}_{\tilde{\mathbf{Q}}}\mathbf{v}^{\tilde{\mathbf{Q}}}$. This vector will be a vector in the image space $\Re^P$ expressed in the old basis $\mathbf{R}$. Now we translate this vector to the coordinates for the new basis $\tilde{\mathbf{R}}$, *i.e.* we apply ${}_{\tilde{\mathbf{R}}}\mathbf{C}_{\mathbf{R}}$. We finally arrive at the vector ${}_{\tilde{\mathbf{R}}}\mathbf{C}_{\mathbf{R}}\,{}_{\mathbf{R}}\mathbf{W}_{\mathbf{Q}}\,{}_{\mathbf{Q}}\mathbf{C}_{\tilde{\mathbf{Q}}}\mathbf{v}^{\tilde{\mathbf{Q}}}$. Since this procedure works for any vector $\mathbf{v}^{\tilde{\mathbf{Q}}}$, we have shown that ${}_{\tilde{\mathbf{R}}}\mathbf{W}_{\tilde{\mathbf{Q}}} ={}_{\tilde{\mathbf{R}}} \mathbf{C}_{\mathbf{R}}\,{}_{\mathbf{R}}\mathbf{W}_{\mathbf{Q}}\,{}_{\mathbf{Q}}\mathbf{C}_{\tilde{\mathbf{Q}}}$.

The notation here can get rather hairy, so let's strip some of it away and make the problem a bit simpler. Suppose we have a linear transformation that maps $\Re^N \rightarrow \Re^N$, and we let $\mathbf{W}$ be the matrix that represents that transformation when vectors are expressed using the standard basis. Now we want to find the matrix that represents the transformation when we transform to a new basis for $\Re^N$. If we let $\mathbf{C}$ be the matrix that changes coordinates from the standard basis to the new basis, then $\mathbf{C}^{-1}$ changes coordinates to the new basis back to the standard basis. But then

$\mathbf{CWC}^{-1}$ represents the transformation in the new basis since it (1) transforms coordinates back to the old basis by applying C^{-1}, (2) performs the linear transformation in the old coordinates, and (3) changes the answer into the coordinate system.

10.7.3 Finding Eigenvectors and Eigenvalues

FOR NOW SEE HIRSCH AND SMALE OR ANOTHER LINEAR ALGEBRA TEXT.

Chapter 11

Ring Networks and Periodic Patterns

11.1 Ring Networks

We start this chapter with a continuation of our investigation into recurrent networks, but now expand beyond the case of two output units to consider so-called ring networks. Ring networks are networks in which the output neurons are arranged along a one-dimensional line and the connectivity is assumed to be distance-dependent, *i.e.* T_{ik} depends only on $i-k$, not on the position i or k. Usually, the connection strength depends purely on the distance $|i-k|$ and $\mathbf{T}$ is symmetric. In many models, the underlying one-dimensional variable is a circular variable, such as the preferred orientation in models of primary visual cortex, the preferred direction of MT neurons, or neurons that keep track of the head direction within the navigation system of rodents. In other situations the underlying system is not circular, but one is interested in the behavior of the system far from the ends. If one assumes that these so-called boundary effects are small, then it is often useful to connect the two ends and consider a ring (**circular boundary conditions**), since the analysis is generally easier.

Many of the underlying intuitions underlying the behavior of ring networks are clearer when we consider the continuous rather than the discrete case. That is, instead of an activity vector $\mathbf{r}$ that is indexed by an integer i, the output pattern is a function of the underlying circular variable θ: $\mathbf{r}(\theta)$. We will go back and forth rather liberally. Most of the analysis that we will derive for ring networks can be applied to two-dimensional (or higher) systems, but we will focus on one-dimensional systems for simplicity.

In a ring network, the entire matrix of connections can be determined from the connection strengths onto any one neuron, since the connections onto any other neuron can be found by a translation around the ring. We say that the system is **translation invariant**. This means that if we specify the first row of the matrix $\mathbf{T}$ then we can find the entries in the other rows by shifting.

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & \dots & T_{1(N-1)} & T_{1N} \\ T_{1N} & T_{11} & T_{12} & \dots & T_{1(N-2)} & T_{1(N-1)} \\ T_{1(N-1)} & T_{1N} & T_{11} & \dots & T_{1(N-3)} & T_{1(N-2)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ T_{13} & T_{14} & T_{1N} & \dots & T_{11} & T_{12} \\ T_{12} & T_{13} & T_{14} & \dots & T_{1N} & T_{11} \end{bmatrix} \tag{11.1}$$

A matrix of this form is called a **circulant matrix**. We will write, $T_{ik} = \mathbf{T}_{c(i-k)}$, where $\mathbf{T}_c$ is the first row vector of $\mathbf{T}$.

Assuming the network given by the connection matrix $\mathbf{T}$ is translation invariant, it follows that the matrix $\widetilde{\mathbf{T}} = (\mathbf{I}-\mathbf{T})^{-1}$ that gives the net transformation performed by the network is also translation invariant. Therefore, $\widetilde{\mathbf{T}}$ is also a circulant matrix.

$\mathbf{T}$ and $\widetilde{\mathbf{T}}$ are also called **convolution matrices**. To see why, consider the vector $\widetilde{\mathbf{T}}\mathbf{r}$.

$$(\widetilde{\mathbf{T}}\mathbf{q})_i = \sum_k \widetilde{\mathbf{T}}_{ik}\mathbf{q}_k = \sum_k \widetilde{\mathbf{T}}_{c(i-k)}\mathbf{q}_k \tag{11.2}$$

This is the discrete, periodic version of the convolution operation we encountered in chapter 2:

$$f * g(t) = \int_{-\infty}^{\infty} ds f(s) g(t-s) \tag{11.3}$$

The effect of multiplying by a convolution matrix $\widetilde{\mathbf{T}}$ is to "smooth" the vector $\mathbf{r}$ by a "window" determined by $\widetilde{\mathbf{T}}_c$.

11.2 Cosine Series

Following our approach from the last chapter, we would like to find an eigenbasis that diagonalizes $\widetilde{\mathbf{T}}$. Luckily, there is a single eigenbasis that diagonalizes every symmetric convolution matrix. This basis is known as a **Fourier basis** and the transformation to the coordinates determined by this basis is known as the **Fourier transform**. Fourier analysis can be applied in situations quite a bit more general than our rather limited setting of symmetric ring networks. For a more systematic treatment, the reader should look at any of a number introductory text books.

The eigenbasis that works is the basis consisting of the vector $\mathbf{1}$ and the discrete sine and cosine functions whose jth component is given by:

$$v_j^{fc} = \cos(2\pi\ f(j-1)/N) \tag{11.4}$$

$$v_j^{fs} = \sin(2\pi\ f(j-1)/N) \tag{11.5}$$

f is the parameter that determines the frequency of the sine or cosine function. ($j-1$ is used instead of j so that the first index corresponds to cos(0) and sin(0) no matter what the frequency.) Since N is the dimension of our vector space we will need a total of N basis vectors. If N is odd, the basis vectors are the $\mathbf{1}$ vector and the sine and cosine function for f ranging from 1 to $(N-1)/2$. If N is even then f goes from 1 to $N/2$, but we don't include the cosine term for $f = N/2$. If we arrange the fourier basis as $\mathbf{F} = \{\mathbf{1}, \mathbf{v}^{1c}, \mathbf{v}^{1s}, \mathbf{v}^{2c}, \mathbf{v}^{2s}, \ldots\}$, then in that basis

$$\widetilde{\mathbf{T}} = \begin{bmatrix} \lambda^0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & \lambda^{1c} & 0 & 0 & 0 & \ldots \\ 0 & 0 & \lambda^{1s} & 0 & 0 & \ldots \\ 0 & 0 & 0 & \lambda^{2c} & 0 & \ldots \\ 0 & 0 & 0 & 0 & \lambda^{2s} & \ldots \\ \vdots & & & & & \end{bmatrix}_{\mathbf{F}} \tag{11.6}$$

Switching to the continuous case, writing a vector as a linear combination of these basis vectors is akin to writing the continuous function

$$q(\theta) = c_0 + \sum_f c_f^c \cos(f\theta) + c_f^s \sin(f\theta) \tag{11.7}$$

The fourier basis breaks the original function apart in terms of it's periodicities. Using trig identities for the sum of a sine and cosine function, we can rewrite

$$q(\theta) = c_0 + \sum_f c_f \cos(f\theta + \phi) \tag{11.8}$$

where $c_f = \sqrt{(c^c_f)^2 + (c^s_f)^2)}$ and $\phi = \arctan(c^s_f / c^c_f)$. c_f represents the amplitude of the modulation at frequency f and ϕ is the **phase**. For symmetric matrices, the sine and cosine basis functions at the same frequency have the same eigenvalue. Therefore, *a symmetric convolution matrix acts to differentially scale the magnitude of the component of the input vector at each frequency.* We say that the transformation is a **linear filter** – the frequencies with large eigenvalues get through (or get amplified) whereas those with small eigenvalues get "filtered out."

11.2.1 The Fourier Transform and the Convolution Theorem

The process of changing coordinates from the standard basis to the fourier basis is known as taking the **fourier transform**. MORE.

11.3 Invariant Tuning and Noise Removal

Now we return to the question of magnitude invariant tuning. In particular, suppose that the connectivity is such that the eigenvalues of $\widetilde{\mathbf{T}}$ are equal to 0 except for $\lambda^{1c} = \lambda^{1s} \neq 0$. Then no matter what the input vector is, the pattern of output activities will be equal to $\lambda^{1c} c_f \cos(f\theta + \phi)$, since all other components of the input are filtered out. While the overall shape of the output activity is determined (a cosine function), it's magnitude $\lambda^{1c} c_f$ and phase ϕ depend on the input. But this is just another way of saying that the output pattern can be written as a product of a cosine-shaped tuning function and a function $\lambda^{1c} c_f$ of the input magnitude, *i.e.* the tuning is magnitude invariant.

A more realistic, nonlinear version of this selective filter model has been proposed to underly contrast-invariant tuning in the visual cortex (**??**). The key elements of this model are that the pattern of recurrent connections is the main determinant of the pattern of output activity, whereas the orientation of the input determines the peak of the pattern, and the magnitude of the input drives changes in the magnitude of the input. This class of models has been described as set up a particular resonant structure so that any input "rings" the network in the same way, but where the ringing is centered and how loud it is depends on the input.

From a different perspective, this network can be seen as eliminating noise. Consider the monkey population coding example (in two dimensions) and arrange the motor neurons according to their preferred direction. As before, the tuning curve of each neuron will have a cosine shape. Thus, adding recurrent connections such that only $\lambda^{1c} = \lambda^{1s} \neq 0$ won't effect the shape of the tuning curve. However, suppose that at some level the neurons are subject to noise, and that this noise is modelled as a fixed, additive component to each neuron's input. Given these assumptions, then the recurrent connections will filter out all components of this noise, except the component of the noise that has a cosine shape. But recall that the fourier basis is an orthogonal basis, so that filtering out the noise corresponds to a perpendicular projection onto the subspace spanned by the vectors $\mathbf{v}^{1c}$ and $\mathbf{v}^{1s}$ (the subspace of cosine-shaped responses). Since there is no way to filter out noise components that look exactly like they are driven by the noiseless input, the recurrent connections perform an optimal filtering of the noise. Again, a more realistic, nonlinear version of this network has been evaluated for it's ability to optimally reconstruct a stimulus (**?**).

Chapter 12

Linear Dynamics

12.1 Introduction

Much of computational neuroscience concerns systems that are dynamic, *i.e.* they change in time. For example, the presentation of a single, static stimulus might trigger an entire chain of neural events – a transient "onset" burst of activity followed by a sustained response that slowly decays due to neuronal adaptation. On a slower time scale, the pattern of and strength of synaptic connections within a given brain region may change, perhaps triggered by developmental processes or as part of a learned response to a set of external stimuli. In fact, the brain is an incredibly complex web of interacting dynamic systems operating on time scales from less than a millisecond to years.

To cope with this dizzying complexity, computational neuroscientists usually focus on a small number of mechanisms operating within a narrow range of time scales, and assume that these mechanisms can be separated out from dynamic processes operating at other time scales. Slower processes are assumed to change so slowly that they can reasonably be viewed as being fixed in time. Faster processes are often assumed to happen frequently enough so that one need only consider their average effect.

Most studies in computational neuroscience focus on one of two basic dynamical problems. The first problem is to understand how patterns of activity are generated by a given neural circuit. The parameters determining the structure of the circuit (number and type of neurons, strengths of synapses, etc.) are viewed as fixed over the time scale of activity. It is also common to average over individual spikes and only consider rates, although so-called spiking networks are also studied. The second basic problem concerns *neural plasticity*, *i.e.* the changes that occur within neural circuits during learning and development. Most commonly, plasticity is assumed to be a slow process, with changes in synaptic strength dependent upon the average coincidence of pre and postsynaptic activity over the course of many stimulus presentations.

12.2 Continuous or Discrete Time?

Before getting into the biological examples, we note that all dynamical systems can be divided into two basic types, depending on whether time is a continuous or discrete variable. In the discrete case, time ticks off in a series of regular intervals or steps. Usually the state of the system at the next time step is some function of the current state of the system. If $\mathbf{x}^{n-1}$ is the state of the system at time step $n-1$, we write

$$\mathbf{x}^n = f(\mathbf{x}^{n-1}) \;=\; f^n(\mathbf{x}^0) \tag{12.1}$$

where f^n denotes n repeated applications of the function f. A **trajectory** of such a system, *i.e.* the points $\{\mathbf{x}^0, \mathbf{x}^2, \mathbf{x}^3, \ldots\}$ looks like an infinite series of points starting at $\mathbf{x}^0$ (figure 12.2, *left*).

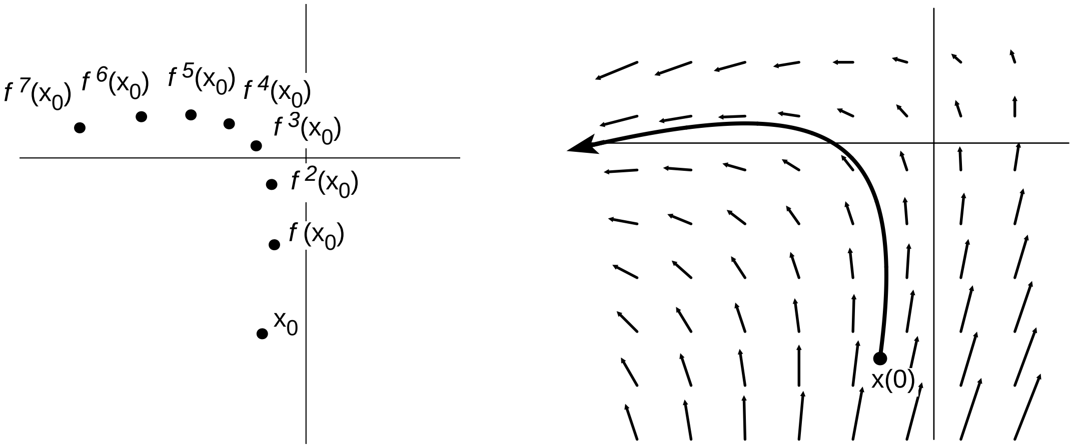


Figure 12.1: Trajectories from discrete (*left*) and continuous (*right*) dynamical systems.

In continuous time dynamical systems, the state of the system evolves smoothly as time flows onward. Usually, the *derivative* of the state depends on the current state of the system, *i.e.*

$$\frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}} = f(\mathbf{x}) \tag{12.2}$$

All the possible values of $\mathbf{x}$ determines the state space of our system, and equation (12.2) assigns a vector $\dot{\mathbf{x}}$ to that point that determines how the system is changing if the system is in the state $\mathbf{x}$. The mapping from states $\mathbf{x}$ to derivative vectors $f(\mathbf{x})$ is known as a **vector field**. In two or three dimensions, this assignment of derivative vectors to points in space can be pictured by sampling the state space at a grid of points $\mathbf{x}$, and then placing a vector with its base at $\mathbf{x}$ (figure 12.1, *right*). Trajectories of the dynamical system are then continuous lines that "follow the arrows." The length of each vector represents how fast the trajectory passes that point. Note that we are using the same two dimensional surface to represent the derivative vectors $\dot{\mathbf{x}}$ and state vectors $\mathbf{x}$, and these have different units. Therefore, the overall scale with which the derivative vectors are drawn is arbitrary.

In these notes, we will only consider examples where time is continuous. Similar (although slightly different) mathematical tools can be used to solve the analogous discrete versions of these examples (see *e.g.* the text by Hirsch and Smale, 1973).

Problems

12.3 Biological Examples

The goal of this chapter is to understand the solutions to these two basic dynamical problems. We begin the chapter by introducing the problems, and derive the linear equations that describe the simplest versions of these problems (the next chapter will introduce the nonlinearities needed to make these examples look a bit more like biology). We will then describe the basic mathematical techniques that can be used to solve linear dynamical systems. We'll then go back and apply these techniques to our examples, and introduce additional mathematical techniques that can be applied to specific versions of these problems.

12.3.1 Activity in a network of connected neurons

Our first example concerns the dynamics in a network of mutually connected neurons or neural populations. The neurons presented here are very similar to the ones in chapter 6, except that the internal state is not instantaneously determined by the summed input, but rather tracks changes in the total input. In particular we assume that the rate of change in the internal state is proportional to difference between the internal state and the current value of the input. So if the current input is large, the internal will increase rapidly, slowing down as it approaches the value of the input until eventually the internal state is equal to the summed synaptic input. Since these dynamics require a distinction between the total input and internal state, we have to introduce a new variable. We will retain the notation that s_i is equal to the total synaptic input to neuron i. We will let u_i denote the internal state of neuron i. Mathematically we write

$$\tau \dot{u}_i \quad = \quad -u_i + s_i \tag{12.3}$$

The time constant τ determines time scale over which the internal state u_i relaxes to the value s_i of the total input. If τ is large the derivative is small (since $\dot{u}_i = (-u_i + s_i)/\tau$) and the decay is slow. τ has units of time, and is usually given in milliseconds. A common interpretation of these equations is that u_i represents an average of the membrane voltage in neuron i, τ is the membrane time constant, and s_i is equal to the total synaptic current times the membrane resistance ($s_i = I^{syn}R$). These issues will be discussed more thoroughly in chapter 5. We'll start by confining ourselves to linear neuron models, we let the input/output function be linear, $g(u_i) = g\ u_i$. As before, we will assume that $g = 1$.

The main architecture that we will consider is the two layer recurrent network introduced at the end of chapter 7. We will assume that neurons in the input layer suddenly change their firing rate to a new pattern, and then remain constant. The focus of our investigation will be to understand how the activity in the output layer reacts to this sudden change in input. The fixed parameters in the network are then the pattern of feedforward weights $\mathbf{W}$, the recurrent weights $\mathbf{T}$ and the pattern of input $\mathbf{q}$. Therefore, the total synaptic input coming in to any output neuron i is given by

$$s_i = \sum_j \mathbf{T}_{ik} g(u_k) + \sum_k \mathbf{W}_{ij} q_j \tag{12.4}$$

The dynamical equations are then

$$\tau \dot{u}_i = -u_i + \sum_k \mathbf{T}_{ik} u_k + \sum_k \mathbf{W}_{ij} q_j \tag{12.5}$$

Using vector and matrix notation

$$\tau \dot{\mathbf{u}} = -\mathbf{u} + \mathbf{T}\mathbf{u} + \mathbf{W}\mathbf{q} \ = \ (-\mathbf{I} + \mathbf{T})\mathbf{u} + \mathbf{W}\mathbf{q} \tag{12.6}$$

If we consider the case with no external input, *i.e.* $\mathbf{q} = 0$, then we have a *linear* differential equation, since the derivative of the state vector $\mathbf{u}$ is a linear function of the current state $\mathbf{u}$.

12.3.2 Development in a set of synapses

The second example concerns development in a set of synapses according to a Hebbian learning rule. In this example we will use the standard trick of averaging over activity patterns (which are assumed to change quickly) to determine the changing pattern of synaptic weights. Note that in this example the dynamic *variables* are the weights and the *parameters* are result from the patterns of activity. In the previous example, the activity levels were the variables, and weights were parameters.

We focus on the synapses onto a single neuron with weight vector $\mathbf{w}$. For each fixed pattern of inputs, the Hebbian learning equation (8.1) can be rewritten in a dynamical form, *i.e.*

$$\dot{\mathbf{w}}_j = \alpha q_j r \tag{12.7}$$

where r is the output of the neuron in question and q_j is the activity in the jth input neuron. Using vector notation,

$$\dot{\mathbf{w}} = \alpha q r \tag{12.8}$$

Because we are constructing a model of the gradual change in synaptic strength occurring during the course of neural development, we will assume that the average change in the pattern of weights will be guided by the average correlation of input and output, *i.e.*

$$\dot{\mathbf{w}} = \alpha \langle \dot{\mathbf{w}} \rangle \;=\; \langle \alpha q r \rangle \tag{12.9}$$

where $\langle x \rangle$ denotes the average value of the quantity x, averaged over a representative sample of input patterns $/rin$.

Again, we will start by examining a linear neuron model. For a linear model, the output $r = \sum w_j q_j = (q)^{\mathrm{T}}\mathbf{w}$. Substituting into equation (12.9), we have

$$\dot{\mathbf{w}} = \langle \alpha q r \rangle \;=\; \alpha \langle q(q)^{\mathrm{T}}\mathbf{w} \rangle \;=\; \alpha \langle q(q)^{\mathrm{T}} \rangle \, \mathbf{w} \tag{12.10}$$

Letting $\mathbf{C}$ be the matrix $\langle q(q)^{\mathrm{T}} \rangle$, we find that we need to solve the following linear differential equation:

$$\dot{\mathbf{w}} = \langle \alpha q r \rangle \;=\; \alpha \langle q(q)^{\mathrm{T}}\mathbf{w} \rangle \;=\; \alpha \mathbf{C}\mathbf{w} \tag{12.11}$$

Thus the equation governing the dynamics of activity within a recurrent network linear neurons are of the same form as the equation governing Hebbian learning of the weights onto a single linear neuron. The mathematical tools needed to solve linear differential equations should be applicable to both problems.

Before going on to develop these tools, let's look at the matrix $\mathbf{C}$ in a little more detail. The ijth entry of $\mathbf{C}$ is given by $C_{ij} = \langle q_i q_j \rangle$, *i.e.* $\mathbf{C}$ is simply the matrix of correlations in the activities of the presynaptic neurons. As we might have expected, the dynamics of a set of synapses developing according the a Hebbian rule is governed by the correlations to be found amongst its inputs.

12.4 One dimensional systems

Now we go on to mathematics of linear dynamical systems. For the next few sections the presentation will be strictly mathematical – we'll need some rather sophisticated tools before returning to solve our example problems. We start by solving a very simple, a one-dimensional dynamical system:

$$\dot{x} = ax \tag{12.12}$$

This equation is easily solved:

$$x(t) = x(0)e^{at} \tag{12.13}$$

Note that the solution $x(t)$ depends on the **initial condition** $x(0)$ as well as on the form of the dynamic equation 12.12.

Now we consider a slightly more difficult equation:

$$\dot{x} = ax + b \tag{12.14}$$

Note that this system is *nonlinear*, since the derivative is no longer a linear function of the state x. We can rewrite the equation as

$$\frac{-1}{a}\dot{x} = -x + \frac{-b}{a} \tag{12.15}$$

As we saw with the passive RC circuit, this equation represents an exponential decay to the value $\frac{-b}{a}$ with a time constant $\tau = \frac{-1}{a}$:

$$x(t) = x(0)e^{-t/(-1/a)} + \frac{-b}{a}(1 - e^{-t/(-1/a)}) \tag{12.16}$$

If $a > 0$ the "time constant" $\tau < 0$, and the system represents exponential growth rather than exponential decay.

$$x(t) = x(0)e^{at} + \frac{-b}{a}(1 - e^{at}) \tag{12.17}$$

This equation can also be solved by considering the new variable $y = x + b/a$, and substituting into equation (**??**) (problem 12.4.1).

Problems

Problem 12.4.1 Show that the substitution $y = x + b/a$ transforms equation (**??**) into a linear equation in y. Then solve for y and substitute back in to obtain the solution given in equation (12.17).

12.5 Stability

12.6 Phase Plane Analysis

12.7 Diagonal Matrices

Now we're going to generalize equation (12.12) and solve the higher dimensional linear dynamical system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{12.18}$$

where $\mathbf{A}$ is some matrix. As usual we'll start with the easy cases an then work up. Suppose that $\mathbf{A}$ is a **diagonal matrix**, *i.e.* $\mathbf{A}_{ij} = 0$ for $i \neq j$. $\mathbf{A}$ looks like

$$\mathbf{A} = \begin{bmatrix} A_{11} & 0 & \dots & 0 \\ 0 & A_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{NN} \end{bmatrix} \tag{12.19}$$

But then the ith component of the vector $\mathbf{Ax}$ is just $A_{ii}x_i$. Then the matrix equation (12.18) is equivalent to a collection of independent equations

$$\dot{x_i} = A_{ii}x_i \tag{12.20}$$

These can be solved as above to yield

$$x_i(t) = x_i(0)e^{A_{ii}t} \tag{12.21}$$

Let's interpret this situation geometrically. Suppose we construct a vector field corresponding to the two dimensional system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ with

$$\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix} \tag{12.22}$$

Then the horizontal component of the vector field is proportional to 2 times the horizontal component of the state $\mathbf{x}$, and the vertical component is proportional to -1 times the vertical component. Figure 12.2 shows three trajectories of this system. The location of the state at times $t = 0, 1, 2$ are represented by the dots along the trajectory. One trajectory starts at the point $[.5, 2]^{\mathrm{T}}$. As expected the horizontal component shows an exponential increase and the vertical component an exponential decay. A second trajectory starts at the point $[.5, 0]^{\mathrm{T}}$. Because the vertical component of this state is 0, the vertical component of the vector field is zero, an the trajectory never leaves the horizontal axis. The corresponding statement is true of the trajectory that starts at the state $[0, 2]^{\mathrm{T}}$: because the horizontal component of this state is 0, the horizontal component of the vector field is zero, and the trajectory never leaves the vertical axis. Note also that because the system is linear, the trajectory with initial condition $\mathbf{x}(0) = [.5, 2]^{\mathrm{T}}$ can be obtained by vector addition of the (straight) trajectories along the horizontal ($\mathbf{x}(0) = [.5, 0]^{\mathrm{T}}$) and $\mathbf{x}(0) = [0, 2]^{\mathrm{T}}$. In fact, if we let $\mathbf{x}_{[.5,0]^{\mathrm{T}}}(t)$ and $\mathbf{x}_{[0,2]^{\mathrm{T}}}(t)$ denote these two special trajectories, then the trajectory starting at $\mathbf{x}(0) = c_1[.5, 0]^{\mathrm{T}} + c_2[0, 2]^{\mathrm{T}}$ is given by $\mathbf{x}(0) = c_1\mathbf{x}_{[.5,0]^{\mathrm{T}}}(t) + c_2\mathbf{x}_{[0,2]^{\mathrm{T}}}(t)$. But since *any* initial condition can be written as a linear combination of the vectors $[.5, 0]^{\mathrm{T}}$ and $[0, 2]^{\mathrm{T}}$, we can use $\mathbf{x}_{[.5,0]^{\mathrm{T}}}(t)$ and $\mathbf{x}_{[0,2]^{\mathrm{T}}}(t)$ to construct any trajectory we want.

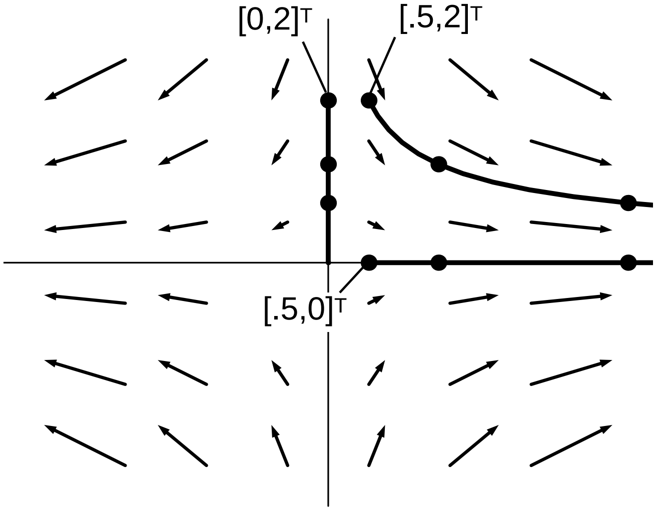


Figure 12.2: Example of a dynamical system defined by a diagonal matrix **A**.

12.8 Eigenvectors

To solve dynamical systems when $\mathbf{A}$ is not diagonal, we need to generalize the procedure outlined at the end of the previous section. The key is finding the initial conditions that lead to straight trajectories. For the trajectory to be straight, the vector field at any point along the trajectory must lie parallel to the vector describing that point. We can write this condition as

$$\dot{\mathbf{x}}(t) = \lambda \mathbf{x}(t) \tag{12.23}$$

where λ is a constant that gives the size of the derivative vector. Negative values of λ yield derivative vectors that are in the opposite direction (but still parallel) to the vector $\mathbf{x}(t)$. For linear dynamical systems, this is equivalent to the condition

$$\mathbf{A}\mathbf{x}(t) = \lambda \mathbf{x}(t) \tag{12.24}$$

This condition captures one of the most important concepts from linear algebra:

Definition 18 *Given a square matrix* $\mathbf{A}$*, a vector* $\mathbf{v}$ *is called an* **eigenvector** *for* $\mathbf{A}$ *if it satisfies* $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$*, for some value of* λ*. The value of* λ *that makes this condition true is called the* **eigenvalue** *of* $\mathbf{A}$ *associated with the vector* $\mathbf{v}$*. Note that any multiple of an eigenvector is also an eigenvector. Therefore, it may be more appropriate to speak of* **eigendirections***. We'll use both terminologies.*

If the initial condition $\mathbf{x}(0)$ is an eigenvector of the matrix $\mathbf{A}$ then the derivative vector will be parallel to $\mathbf{x}(0)$, and hence the entire trajectory will lie in the eigendirection defined by $\mathbf{x}(0)$. For such an initial condition, the system behaves like a one-dimensional system and the trajectory

$$\mathbf{x}(t) = \mathbf{x}(0)e^{\lambda}t \tag{12.25}$$

This suggests that for any matrix $\mathbf{A}$,if we could express every vector as a linear combination of eigenvectors, then solving the system $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$ is easy. Let $\{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^N\}$ denote the set of eigenvectors and $\{\lambda^1, \lambda^2, \ldots, \lambda^N\}$ denote the corresponding set of eigenvalues. Then if we are given an initial condition $\mathbf{x}(0)$, we first write that vector as a linear combination of eigenvectors:

$$\mathbf{x}(0) = c_1\mathbf{v}^1 + c_2\mathbf{v}^2 + \ldots + c_N\mathbf{v}^N \tag{12.26}$$

But since we know the trajectories for these eigenvectors and we are dealing with a linear system, we can simply write down the solution:

$$\mathbf{x}(t) = c_1\mathbf{v}^1 e^{\lambda^1 t} + c_2\mathbf{v}^2 e^{\lambda^2 t} + \ldots + c_N\mathbf{v}^N e^{\lambda^N t} \tag{12.27}$$

12.8.1 Eigenbases

Assuming that any initial condition $\mathbf{x}(0)$ can be written as a linear combination of eigenvectors $\{\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^N\}$, is (by definition) equivalent to assuming that the eigenvectors form a basis $\mathbf{V}$ for the state space. We call such a basis of eigenvectors, $\mathbf{V}$, an **eigenbasis**. Now suppose we change coordinates to this eigenbasis. What does our matrix $\mathbf{A}$ look like? Let's start by expressing the eigenvector $\mathbf{v}^1$ in the basis $\mathbf{V}$: $\mathbf{v}^1 = [1, 0, \ldots, 0]^{\mathrm{T}}$. Since $\mathbf{v}^1$ is an eigenvector, we have

$$\mathbf{A}\mathbf{v}^1 = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \ldots & A_{NN} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{N1} \end{bmatrix} = \begin{bmatrix} \lambda^1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \lambda\mathbf{v}^1 \tag{12.28}$$

Therefore, $A_{11} = \lambda^1$ and $A_{i1} = 0$ for $i > 1$. Making this calculation for each eigenvector we see that $\mathbf{A}$ *is a diagonal matrix when expressed in the eigenbasis* $\mathbf{V}$. In other words, assuming we can find a set of eigenvectors that span the entire state space, we can simply change coordinates to this basis and solve the simple case of a linear dynamical system defined by a diagonal matrix.

Note that not all matrices have such a set of eigenvectors. Luckily, one can prove that *if* $\mathbf{A}$ *is a symmetric matrix* ($\mathbf{A} = \mathbf{A}^{\mathrm{T}}$) *then* $\mathbf{A}$ *does have a complete basis of eigenvectors.* But things are even nicer than this. Not only can we find a complete basis, we can find an *orthonormal* basis of eigenvectors. Thus, changing coordinates to such a basis is easy – we just need to project on the different eigenvectors. We will use these facts in the next two chapters.

Chapter 13

The Dynamics of Hebbian Development

13.1 Development in a Set of Synapses - Ocular Dominance

Now we have the tools to return to our biological examples. Let's reconsider equation (12.11):

$$\dot{\mathbf{w}} = \langle \alpha q r \rangle \; = \; \alpha \left\langle q(q)^{\mathrm{T}} \mathbf{w} \right\rangle \; = \; \alpha \mathbf{C} \mathbf{w}$$

To solve this, we'd like to find the eigenvectors for the correlation matrix $\mathbf{C}$. Luckily, $\mathbf{C}$ is a symmetric matrix, so it does have an orthonormal basis of eigenvectors. Actually, because $\mathbf{C}$ is a correlation matrix, *i.e.* $\mathbf{C} = \langle q(q)^{\mathrm{T}} \rangle$, it can be proved that all the eigenvalues of $\mathbf{C}$ are non-negative ($\lambda_i \geq 0$). Figure 13.1 shows the trajectory equation (12.11), where the two dashed lines represent the eigendirections of the system. The eigenvalue λ^1 in the direction running mostly horizontally is equal to 2, and the eigenvalue λ^2in the direction running mostly vertically is equal to 1. The left plot shows the trajectory for small values of t, when the $\mathbf{x}(t)$ remains near the initial condition $\mathbf{x}(0)$. The initial condition was chosen to have equal magnitude in each of the eigendirections. The right plot shows the same trajectory at 1/10 the magnification. As t grows large, the exponential growth in the horizontal component begins to dominate. If we take the ratio of the horizontal to vertical components we find

$$x_1(t)/x_2(t) = e^{\lambda^1 t}/e^{\lambda^2 t} = e^{(\lambda^1 - \lambda^2)t} \tag{13.1}$$

i.e. the dominance of the horizontal component grows exponentially.

How can we use figure 13.1 to inform the biology? First we must address the obvious, non-biological aspects of our model. Most obviously, the weights are becoming infinitely large. This is a consequence of the fact that Hebbian learning is a positive feedback system: making a connection stronger, makes coincident activity in the pre and post-synaptic neurons more likely, which further increases the strength of the connection. Instead of addressing this issue directly, we will simply assume that biology has come up with some mechanisms to keep the weights from growing out of control. Thus, at some point, the trajectory will cease growing. We further assume that this process doesn't otherwise alter the pattern of weights in any significant manner. The qualitative picture to take home from figure 13.1 is that if the system starts with initially small connections, then correlation-based learning rules will settle on weight vectors that lie close to the direction of the eigenvector of the correlation matrix $\mathbf{C}$ with the largest eigenvalue.

13.1.1 Principal Components Analysis

Given a distribution of vectors and the corresponding correlation matrix, the eigenvector that has the largest eigenvalue is know as the **principal component** of the distribution. WILL GO OVER PCA IN CLASS.

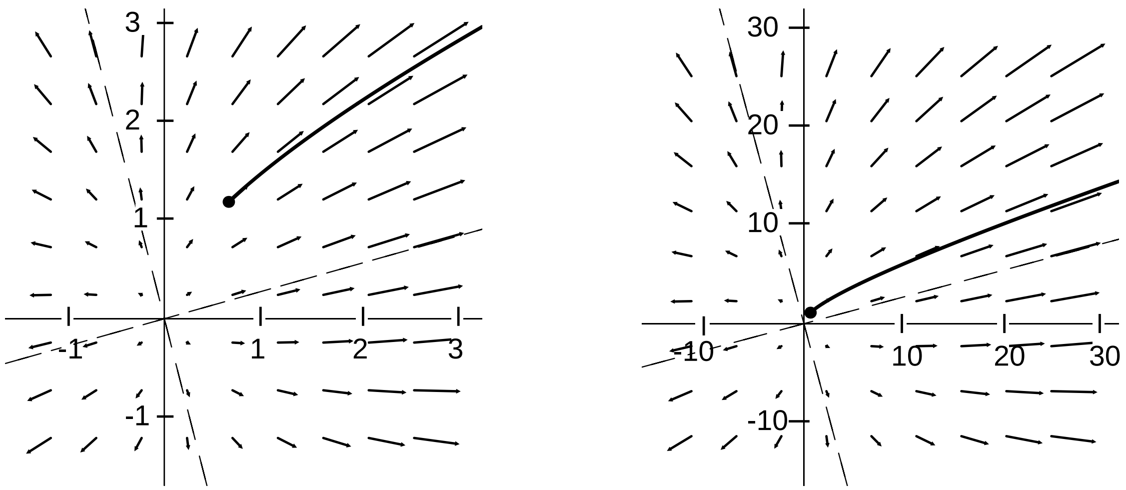


Figure 13.1: Example of a dynamical system with $\lambda^1 = 2$ and $\lambda^2 = 1$. Figure at left is at 10 times the magnification of figure at left.

13.1.2 A Simple Model

Now that we have some of the mathematical tools to solve linear differential equations, let's apply them to a simple example of synaptic plasticity – the development of **ocular dominance**. Ocular dominance refers to the fact that while many cells in the visual cortex receive inputs from both eyes, the degree that cells receive more input from one eye (ocular dominance) varies across cells. In fact, in many species ocular dominance is **mapped** on the cortex, *i.e.* cells with similar ocular dominance are found close to each other. We'll address the issue of mapping later. First we'll consider a very simple example.

Suppose we examine a single cell in the primary visual cortex (V1), and look at the average input from two populations of input neurons in the LGN, one population represents LGN neurons getting input from the left eye and the other represents the population of LGN neurons getting input from the right (figure 13.2, *left*). We let

$$\mathbf{C} = \begin{bmatrix} 1 & \epsilon \\ \epsilon & 1 \end{bmatrix} \tag{13.2}$$

be the matrix of correlations between LGN cells receiving input from the two eyes. The ones along the diagonal mean that we are assuming that the mean squared activity in each LGN population is equal to one, in whatever abstract units we are representing these correlations. The ϵ in the off diagonal position means that the correlation between activity in the two populations is ϵ times as strong as the mean squared activity in each population. If ϵ is positive, activity in the two eyes are positively correlated; if ϵ is negative, activity in the two eyes are negatively correlated. Note that since the correlations between eyes can be no stronger than the correlations of a single population with itself, we have that $-1 \leq \epsilon \leq 1$.

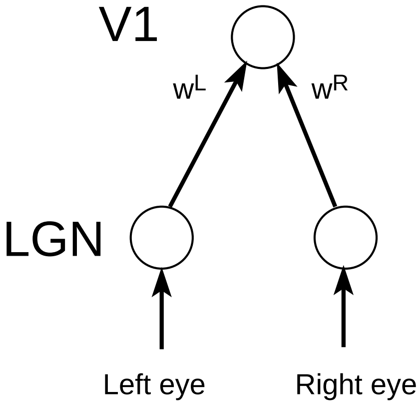


Figure 13.2: A simple model of ocular dominance development.

Let's find the eigenvalues and eigenvectors of $\mathbf{C}$. While there is a system for finding eigenvalues and eigenvectors, one time honored strategy for solving problems in mathematics is to guess the solution and then prove that you are right. This is an especially useful strategy when there are important symmetries in the problem. In the problem at hand, we have made no distinction between the right-eye and left-eye LGN populations, so we'd expect that the eigenvectors shouldn't be changed if we switched w_L and w_R. This actually confines our guesses quite a bit in our two dimensional example, since there are only two directions in the plane that are unchanged if we switch axes (switching axes is equivalent to reflecting the plan around the 45^o line where $w_L = w_R$). The two directions are the 45^o line itself, and the line perpendicular to it. Thus, good guesses for eigenvectors would be the vector $[1,1]^{\mathrm{T}}$ and $[1,-1]^{\mathrm{T}}$. Multiplying by the matrix $\mathbf{C}$ we find that

$$\mathbf{C}[1,1]^{\mathrm{T}} = (1+\epsilon)[1,1]^{\mathrm{T}} \tag{13.3}$$

$$\mathbf{C}[1,-1]^{\mathrm{T}} = (1-\epsilon)[1,-1]^{\mathrm{T}} \tag{13.4}$$

So these vectors are indeed eigenvectors. The corresponding eigenvalues of $1+\epsilon$ and $1-\epsilon$. Note that if we wanted to consider ortho*normal* eigenvectors, then we would use the unit vectors $[1,1]^{\mathrm{T}}/\sqrt{2}$ and $[1,-1]^{\mathrm{T}}/\sqrt{2}$.

The biological interpretation of this eigenbasis is rather easy. First, the eigendirection $[1,1]^{\mathrm{T}}$ represents the sum of w_L and w_R, *i.e.* this direction represents the total synaptic weight onto our V1 neuron. The eigendirection $[1,-1]^{\mathrm{T}}$ represents the difference between w_L and w_R, with positive values meaning that the contribution from the left eye is dominant and negative value meaning that the right eye is dominant. A reasonable definition of ocular dominance (OD) is the difference between the two weights normalized by the sum, *i.e.* $OD = w_L - w_R/(w_L + w_R)$. The fact that the two eigenvalues are perpendicular means that these two components – the sum of the weights and their difference – develop independently. For example, the growth in the difference between w_L and w_R depends only on the current difference – the difference grows just the same if $w_L = 1$ and $w_R = 2$ or if $w_L = 101$ and $w_R = 102$. Likewise, the total weight grows the same whether $w_L = 10$ and $w_R = 10$ or $w_L = 1$ and $w_R = 19$.

In looking at the qualitative behavior of this system, the first thing to note is that since $|\epsilon| \leq 1$ the eigenvalues $1 \pm \epsilon \geq 0$. Therefore, *both* the total input and the ocular dominance are increasing exponentially (except in the non-biological case when the two eyes are exactly correlated or anti-correlated $|\epsilon| = 1$). There are two main cases to consider, depending on whether ϵ is bigger or smaller than 0. If $\epsilon > 0$, the eigenvalue $1 + \epsilon$ is the principal eigenvalue, and the growth in the system is dominated by the growth in the total weight. If $\epsilon < 0$, the eigenvalue $1 - \epsilon$ is the principal eigenvalue, and the growth in the system is dominated by the growth in the difference between the weights. Since we have defined ocular dominance to be the difference component divided by the sum component, we reach the following conclusion from this simple model: *ocular dominance develops only in the case where activity in LGN neurons receiving input from the two eyes is anti-correlated* ($\epsilon < 0$)*; correlated activity in the two eyes* ($\epsilon > 0$) *leads to synaptic connections that have similar strengths, i.e. the ocular dominance is small.*

It is very important to remember that the applicability of this conclusion to biology rests on the assumptions put into the model. The linear model is very simple, and one must always worry about negative activities and weights. Also, the system is unstable in the sense that the total weight will grow infinitely large, unless some other mechanism is incorporated into the model. However, the simplicity of the model can also be an advantage. Since there are very few elements in the model, the relationship between the correlation structure of the input and the qualitative behavior of the model is quite clear. This focuses attention on any new mechanisms added to more complicated models that show qualitatively different behavior. In particular, since animals do develop ocular dominance, our simple model indicates that simple associational rules by themselves are not enough. We should focus attention on biological mechanisms that lead to *competition* between the inputs from LGN cells corresponding to the left and right eyes.

Mathematical Aside. The use of the term correlation is often quite sloppy, and it is sometimes unclear exactly what mathematical calculation the author is referring to. One common confusion when speaking of the correlation between two patterns is whether the mean value is assumed to have been subtracted or not, *i.e.* whether the correlation between vectors $\mathbf{p}^2$ and $\mathbf{p}^2$ is calculated as $\sum_i p_i^1 p_i^2 = \mathbf{p}^1 \cdot \mathbf{p}^2$ or as $\sum_i (p_i^1 - \langle \mathbf{p}^1 \rangle)(p_i^2 - \langle \mathbf{p}^2 \rangle) = (\mathbf{p}^1 - \langle \mathbf{p}^1 \rangle) \cdot (\mathbf{p}^2 - \langle \mathbf{p}^2 \rangle)$ where we have use $\langle \mathbf{p}^k \rangle$ to denote the average value of the entries in the vector $\mathbf{p}^k$. The correlation with the mean subtracted is properly termed the **covariance** of the two patterns. Sometimes it is implicitly assumed that the means have already been subtracted. In this case the correlation is the same as the covariance.

Biological Aside. The reader should note how the circles in figure 13.2 take on various meanings. In this example, it is more reasonable to assume that the LGN "neurons" actually represent *populations* of neurons, and w_L and w_R represent the average synaptic strength from these populations. In contrast, nothing is gained by interpreting the developmental dynamics within the cortical neuron as representing a population of V1 neurons.

13.2 Competition and Subtractive Normalization

So far our simple model has two problems: the weights can grow infinitely large, and ocular dominance does not develop unless the activity in the two eyes is anti-correlated. We can actually make substantial progress on both of these issues at once if we posit a biological mechanism that constrains the total synaptic strength onto a neuron to remain within a reasonable range.

13.2.1 Hard Constraints

The simplest way to do this is to simply assume that the sum of the synaptic strengths onto a neuron is kept fixed by some monitoring mechanism within the cell. Geometrically this means that we are confining our dynamics to remain in the subspace where $\sum_j w_j$ is equal to some constant. In our simple two dimensional example, this so-called **constraint surface** is just a line (figure 13.3). Enforcing this constraint is quite easy to do within our simple model: we simply set the derivative to 0 in the eigendirection corresponding to the sum of w_L and w_R. This corresponds to projecting the vector field onto the constraint surface (in this case its just a line), and examining the dynamics within that surface. Because this surface corresponds exactly to an eigendirection, this is easy. If we let $w_D = w_L - w_R$ denote the difference in the weights, we know that $w_D(t) = w_D(0)e^{(1-\epsilon)t}$. Since $\epsilon < 1$ unless activity in the eyes is identical (in which case $\epsilon = 1$), constraining the sum of the activity has the effect of leading to ocular dominance segregation in all cases.

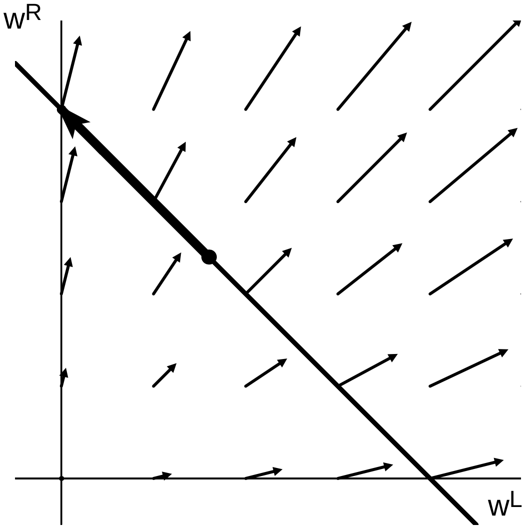


Figure 13.3: Hebbian development with a projection (subtractive) constraint and $\epsilon = 0.5$.

How do we understand this result? SUBTRACTIVE CONSTRAINT. MORE.

13.3 Ocular Dominance Maps

Now we consider the development of ocular dominance in a whole sheet of interacting cortical cells. Primary visual cortex is a largely two dimensional piece of tissue, having an area of several square centimeters, but being only 2 mm in depth (cortex comes from the greek word for bark). Furthermore, electrophysiological recordings reveal that the response properties of cells throughout the thickness of cortex but at the same location. Thus, each “cell” in our developmental models may just as well represent one of these cortical columns, and many models consider the cortex to be two dimensional. To simplify the problem further, we will consider a reduced, one dimensional cortex. This will allow us to explore the importance of spatial structure in the intracortical connectivity, yet still view cortical activity patterns as vectors in a natural way.

So we consider a two layer recurrent network, where the input layer still has just two populations (left eye and right eye) but where the cortex has N neurons or columns. The recurrent connections between the cortical columns are given by the weight matrix $\mathbf{T}$, *i.e.* cortical column j is connected to cortical column i with connection strength $\mathbf{T}_{ij}$. As in chapter 7, given an input vector $\mathbf{q}$, the final activity pattern $\mathbf{r}$ satisfies the following equation:

$$\mathbf{r} = \mathbf{Tr} + \mathbf{Wq} \tag{13.5}$$

Solving this equation we have

$$\begin{aligned} (\mathbf{I} - \mathbf{T})\mathbf{r} &= \mathbf{Wq} && (13.6) \\ \mathbf{r} &= (\mathbf{I} - \mathbf{T})^{-1}\mathbf{Wq} && (13.7) \end{aligned}$$

Since we have two input populations and N output neurons, $\mathbf{W}$ is a $2 \times N$ matrix where the first column represents the weights from the left-responsive LGN population, and the second column represents the weights from the left-responsive LGN population. We will denote these column vectors $\mathbf{w}_L$ and $\mathbf{w}_R$. Note that we have assumed that the matrix $\mathbf{I} - \mathbf{T}$ is invertible. Since its inverse will come up over and over again, we will give it a new name, $\mathbf{B} = (\mathbf{I} - \mathbf{T})^{-1}$ so that $\mathbf{r} = \mathbf{BWq}$. The matrix entry $\mathbf{B}_{ij}$ captures the net effective connectivity from cortical column j to i. We will assume that these connection strengths are distance dependant, *i.e.* the magnitude of $\mathbf{B}_{ij}$ depends only on $|i - j|$. One common assumption is that nearby cortical columns excite each other, while columns at a further distance display mutual inhibition (figure 13.4, *left*). To avoid the different patterns of activity displayed by the columns at the "edge" of our one dimensional line of columns, we assume **circular boundary conditions**, *i.e.* we assume column 1 is right next to column N. By joining the ends of our cortical line in this way, we are considering a so-called **ring network** (figure 13.4, *right*).

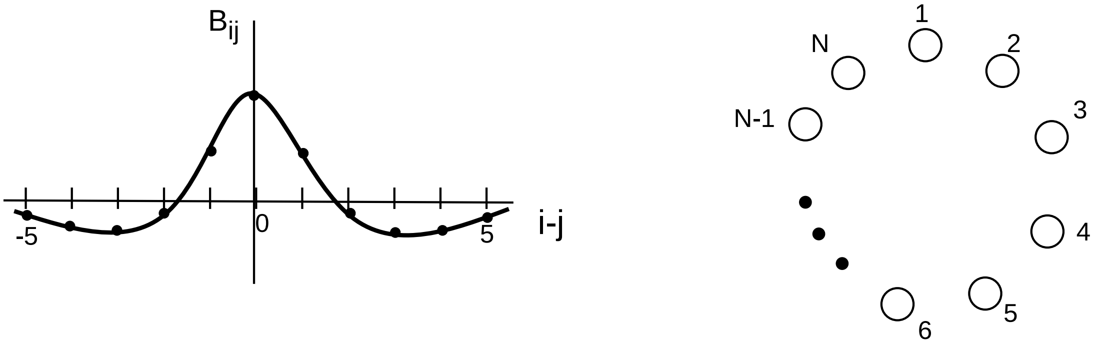


Figure 13.4: Distance-dependent connectivity in a ring network.

Now we examine the learning dynamics

$$\begin{aligned} \dot{W} &= \langle \mathbf{r}\mathbf{q}^{\mathrm{T}} \rangle && (13.8) \\ &= \langle \mathbf{BWq}\mathbf{q}^{\mathrm{T}} \rangle && (13.9) \\ &= \mathbf{BWC} && (13.10) \end{aligned}$$

where $\mathbf{C}$ is the 2×2 correlation matrix $\langle \mathbf{q}\mathbf{q}^{\mathrm{T}} \rangle$. How is equation (13.3) related to our traditional linear dynamical system that comes in the form $\dot{\mathbf{x}} = \mathbf{Ax}$? First of all, equation (13.3) *is* a linear differential equation in the sense that $\dot{W}$ is a linear function of $\mathbf{W}$ (problem 13.3.1). The major difference is that instead of the elements of the vector $\mathbf{w}$ defining an N dimensional state space, the elements of the $N \times 2$ matrix $\mathbf{W}$ defines an $2N$ dimensional space of connection strengths (in general $\mathbf{C}$ will be a $P \times P$ matrix and $\mathbf{W}$ will be NP dimensional).

Like any linear differential equation, our task is to find eigendirections in this space, *i.e.* we must find find specific matrices $\mathbf{W}$ such that $\mathbf{BWC} = \lambda\mathbf{W}$. One might guess that the eigendirections for $\mathbf{W}$ might depend on the eigenvectors for each of the matrices $\mathbf{B}$ and $\mathbf{C}$. In fact, suppose that $\mathbf{u}$ is an eigenvector of $\mathbf{B}$ with eigenvalue γ and that $\mathbf{v}$ is an eigenvector of $\mathbf{C}^{\mathrm{T}}$ with eigenvalue α. The following shows that the outer product $\mathbf{W} = \mathbf{u}\mathbf{v}^{\mathrm{T}}$ is one of the eigendirections that we are looking for, and it has eigenvalue $\gamma\alpha$:

$$\mathbf{BWC} = \mathbf{B}(\mathbf{u}\mathbf{v}^{\mathrm{T}})\mathbf{C} = (\mathbf{Bu})(\mathbf{C}^{\mathrm{T}}\mathbf{v})^{\mathrm{T}} = (\gamma\mathbf{u})(\alpha\mathbf{v})^{\mathrm{T}} = \gamma\alpha\mathbf{W} \tag{13.11}$$

If $\mathbf{B}$ has N independent eigenvectors and $\mathbf{C}^{\mathrm{T}}$ has P, then this process yields the NP eigendirections that can form an eigenbasis for our state space. *Remember, the dynamics develops independently along each of these eigendirections.*

Now if we examine the problem of ocular dominance, the eigenvectors of $\mathbf{C}$ are just those discussed in section 13.1.2 – they represent the sum and difference of the weights going to each of the cortical columns. Suppose we constrain the sum of the weights onto each cortical column to be fixed as before. Then we are "freezing" the dynamics along all eigendirections in weight space that include the sum eigendirection of $\mathbf{C}$. Therefore, we are left with the N dimensional subspace of weight space consisting of weight matrices $\mathbf{W} = \mathbf{w}_{diff}[1, -1]^{\mathrm{T}}$. $\mathbf{w}_{diff}$ is the N dimensional vector whose jth element describes the difference between how strongly the left and right eye are connected to the jth cortical column. Under our constraint, the Hebbian learning equation () should lead to the development of weight matrices that are close to $\mathbf{w}^1_{diff}[1, -1]^{\mathrm{T}}$, where $\mathbf{w}^1_{diff}$ is the principal eigenvector of the matrix $\mathbf{B}$.

13.3.1 Fourier Basis

We have assumed that the intracortical interactions given by the matrix $\mathbf{B}$ are distance dependent, *i.e.* the value B_{ij} depends only on the difference $i - j$. In this case $\mathbf{B}$ is known as a **circulant matrix**. It can be shown that the collection of discrete Fourier vectors form an eigenbasis for all circulant matrices. For N dimensional vectors, the Fourier vectors are given by the formulas

$$v_j = \cos(2\pi kj/N) \tag{13.12}$$

or

$$v_j = \sin(2\pi kj/N) \tag{13.13}$$

k represents the frequency of the discrete vector, since it determines the number of times the argument $2\pi kj/N$ traces out a circle as j goes from 1 to N. MORE. PROBLEMS? Therefore, by performing a discrete Fourier transform on the cortical interaction function (like that shown on the left of figure 13.4), we can find the component $\mathbf{B}$ with the largest eigenvalue. This direction represents the dominant periodicity at which the features represented by the eigenvectors of $\mathbf{C}$ will modulate as one moves across the cortex. For example, the dominant frequency of the center-surround connectivity shown in figure 13.4 will be approximately twice the width of the center hump. In our ocular dominance example, the weights will be such that the strength of the difference between left and right-eye connectivity waxes and wanes with this periodicity.

13.3.2 Receptive Field Development

In the simple examples considered so far, the input from the LGN had no spatial extent – we lumped all cells getting input from the same eye into one either a "left" or "right" population. Perhaps in a more realistic model of ocular dominance, a cortical cell would come to respond to inputs from one eye in one portion of the cell's receptive field, while responding to inputs from the other eye in a different portion of the receptive field. At each location in the LGN, inputs would come from one eye, but over the whole receptive field the cell would be binocular.

It is easy to extend the framework we have developed so far to consider an LGN with a spatial extent. We consider two one-dimensional "rings" of LGN cells, one for the left eye and one for the right. We will assume that each contains P populations. Input vectors will be $2P$ dimensional and we will make the convention that the first P elements of the input vector $\mathbf{q}$ will represent the activity $\mathbf{q}^L$ in the P left-eye populations and the last P elements will represent the activity $\mathbf{q}^R$ in the P right-eye populations, *i.e.*

$$\mathbf{q} = \begin{bmatrix} \vdots \\ \mathbf{q}^{left} \\ \vdots \\ \vdots \\ \mathbf{q}^{right} \\ \vdots \end{bmatrix} \tag{13.14}$$

But then the correlation matrix $\mathbf{C}$ takes the following form

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}^{LL} & \mathbf{C}^{LR} \\ \mathbf{C}^{RL} & \mathbf{C}^{RR} \end{bmatrix} \tag{13.15}$$

where $\mathbf{C}^{LR} = \mathbf{C}^{RL} = \mathbf{C}^{opp}$ represents the $P \times P$ matrix of correlations between right and left eye populations, and $\mathbf{C}^{LL} = \mathbf{C}^{RR} = \mathbf{C}^{same}$ represents the $P \times P$ matrix of within-eye correlations.

Now we use our usual trick of expressing the inputs in sum and difference coordinates. Letting $\mathbf{q}^{sum} = \mathbf{q}^L + \mathbf{q}^R$ and $\mathbf{q}^{diff} = \mathbf{q}^L - \mathbf{q}^R$, we see that the sum and difference are decorrelated:

$$\left\langle \mathbf{q}^{sum} \mathbf{q}^{diff\,\mathrm{T}} \right\rangle = \left\langle (\mathbf{q}^L + \mathbf{q}^R)(\mathbf{q}^L - \mathbf{q}^R)^{\mathrm{T}} \right\rangle = \mathbf{C}^{LL} - \mathbf{C}^{LR} + \mathbf{C}^{RL} - \mathbf{C}^{RR} = 0 \tag{13.16}$$

Therefore, in these new coordinates, $\mathbf{C}$ becomes

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}^{sum} & 0 \\ 0 & \mathbf{C}^{diff} \end{bmatrix} \tag{13.17}$$

where $\mathbf{C}^{sum} = 2(\mathbf{C}^{LL} + \mathbf{C}^{LR})$ and $\mathbf{C}^{diff} = 2(\mathbf{C}^{LL} - \mathbf{C}^{LR})$. Again, the sum and difference components develop independently. Assuming that the sum of the weights is constrained, we will focus on $\mathbf{W}^{diff}$, the $N \times P$ dimensional matrix that holds the difference in w^L and w^R from each of the P LGN locations in the ring to the N cortical locations. We will also assume that input correlations depend on distance, so that $C^{diff} = C^{diff}(|i - j|)$.

As we derived above, the eventual pattern of weights will be dominated by the the outer product principal eigenvector of the cortical interaction matrix $\mathbf{B}$ and the principal eigenvector of C^{diff}. But since we are assuming that correlations depend only on distance in the LGN, C^{diff} is a circulant matrix as well, and it's eigenvectors are given by the discrete Fourier vectors. The Fourier vector with the largest eigenvalue will determine the receptive field for each of the cortical cells. Note that we are actually looking for receptive fields that are very boring, *i.e.* if cells are to be monocular, then the cell must receive input from a given eye over the entire receptive field. Therefore, monocular receptive fields require that the zero frequency component – the vector of ones time the average correlation between LGN populations – be dominant. One can show that this will be the case whenever the correlation function is strictly positive.

Problems

Problem 13.3.1 Show that equation (13.3) defines a linear dynamical system.

Problem 13.3.2 Assume that $\mathbf{B}$, $\mathbf{W}$, and $\mathbf{C}$ are 2×2 dimensional matrices:

$$\mathbf{B} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \ \mathbf{W} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Rewrite $\mathbf{W}$ as a 4 dimensional vector $\mathbf{w}$, and then rewrite equation (13.3) in the more standard form $\dot{\mathbf{w}} = \mathbf{A}\mathbf{w}$.

13.4 Orientation Selectivity

The exact same machinery that we developed to look at ocular dominance can be used to examine the development of orientation selectivity in the visual cortex. Cells tuned for the orientation of a including contrast edges (*e.g.* orientated bars) are first seen at the level of the visual cortex. In the retina and the LGN, most cells are not tuned for orientation, but have so-called center-surround receptive fields, like those in the limulus eye. Some cells respond to light in the center and dark in the surround (ON-center cells), and others to the opposite pattern (OFF-center cells). The dominant hypothesis for the construction of oriented cells was first proposed by Hubel and Wiesel in the early sixties. They proposed that cortical cells receive inputs aligned in such a way so that cortical cells have alternating ON and OFF subregions (see figure 13.5). Moreover, they also showed that the orientation selectivity arising from such a relationship was mapped in the cortex, *i.e.* nearby cortical cells had similar orientation preferences.

Important features of the development of ON and OFF subregions can be modeled by simply substituting ON and OFF for left and right in the above derivation. Oriented cells will develop whenever the dominant eigenvector of the correlation matrix $\mathbf{C}^{diff}$ is anything *other* than the zero frequency component. In this case, cortical receptive fields will contain an oscillation of ON and OFF subregions, and will be orientation selective. As before, the intracortical interaction matrix $\mathbf{B}$ determines the periodicity of orientation selectivity, and hence the structure of the orientation map.

13.5 Localized Receptive Fields

We have shown that a simple framework yields insight into the possible mechanisms underlying the development of both ocular dominance and orientation maps in the visual cortex. The main prediction of these models is that for ocular dominance to develop, the principal eigenvector of matrix obtained by adding the within eye correlations and the negative of the between eye correlations should be the zero frequency vector. For orientated cells, the correlation of the corresponding

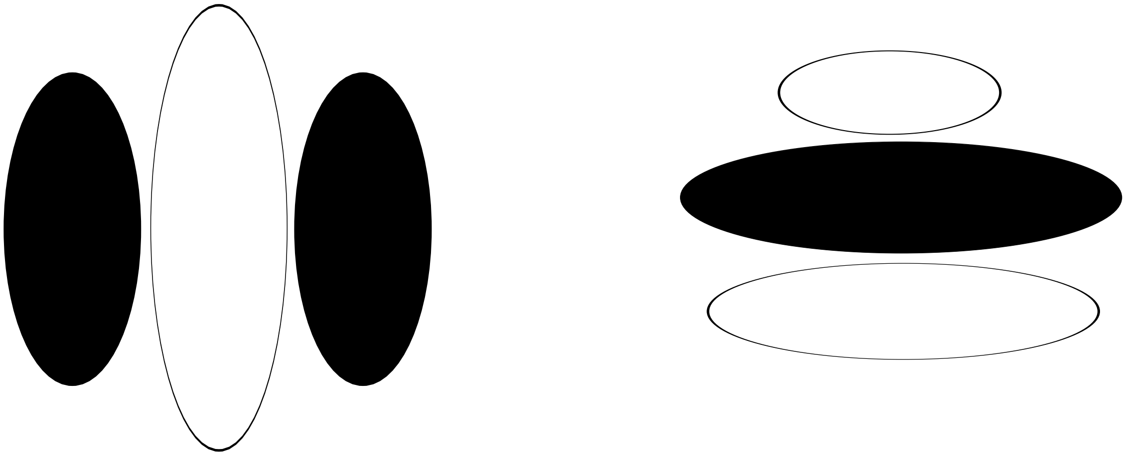


Figure 13.5: Example of simple cell receptive fields for cells tuned to vertical (*left*) and horizontal (*right*) stimuli.

ON-OFF correlation matrix should be dominated by a vector *other* than the zero frequency vector. To the degree that these have been measured accurately, experiments tend to support these hypotheses.

However, there is one major flaw in the simple models presented so far: the connectivity that develops is not local, *i.e.* cortical cells can receive input from LGN responding to any visual location. Conversely, LGN cells can project to all visual cortical neurons. An easy way to remedy this non-biological behavior is to alter the learning equations to account for the fact that it may be quite difficult for cells to develop the large axonal or dendritic arbors that would be necessary to support such global connectivity. One easy way to do this is to assume that each LGN cell has an **arbor function**, that makes it easiest for connections grow toward a specified region of visual cortex. SEE MILLER ARTICLE.

Chapter 14

Attractor Networks

14.1 Memories as Attractors

One of the most important ideas contributed by computational neuroscience is the idea of a memory being stored as an attracting state for the underlying dynamics in a neural network. The groundwork for this idea was part of Hebb's famous book, *The Organization of Behavior* (1949). In this book, Hebb talked about how the plasticity rule that now bears his name could be used to form **cell assemblies**, or groups of interconnected neurons. Because these neurons could sustain activity even when there is no external stimulus present, activity patterns within such assemblies could form the neurological substrate of "on-line" or **working memory**.

> Historical Aside. Hebb's ideas played an important role in returning the "mental" or "cognitive" aspects of behavior as legitimate questions for scientific study. In the earlier half of the century, behaviorist or stimulus-response ideas were dominant in Psychology, especially in the United States. All behavior was assumed to be in response to external stimuli, and it was deemed impossible to "look inside" the brain to study "thinking." After all, the only things that were available for scientific study were the nature of the stimuli that an animal encountered and the subsequent behavioral response. In presenting a clear picture of how neural activity could be generated and sustained, even in the absence of external stimuli, Hebb's work made an important contribution to the increasing study of internal representation and the cognitive abilities of neural circuits.

With the rise of biologically-inspired models of computation in the latter half of the century, the idea of the cell assembly found natural mathematical correlate in the notion of an **attractor**. An attractor is simply a state of a dynamical system such that trajectories that start at nearby states flow toward the attracting state. The simplest example of an attractor is a stable equilibrium (figure 14.1, *left*). Dynamical trajectories can also be attractors, such as the attracting periodic trajectory or **limit cycle** shown in figure 14.1, *right*).

14.2 Energy Functions

One common way to demonstrate that a dynamical system has attractors is to construct an **energy function** or **Lyapunov function** for that system.[1] Consider a mapping that attaches a number to every location in state space. Such a function will be an energy function for a given dynamical system if the "energy" (value of the function) is decreasing along every trajectory of the dynamics. We use the term energy function in analogy with physical systems where the energy within a system does not increase, but can possibly decrease due to dissipative forces such as friction.

---

[1] The term "Lyapunov function" is more common with mathematicians. We will use the more common "energy function."

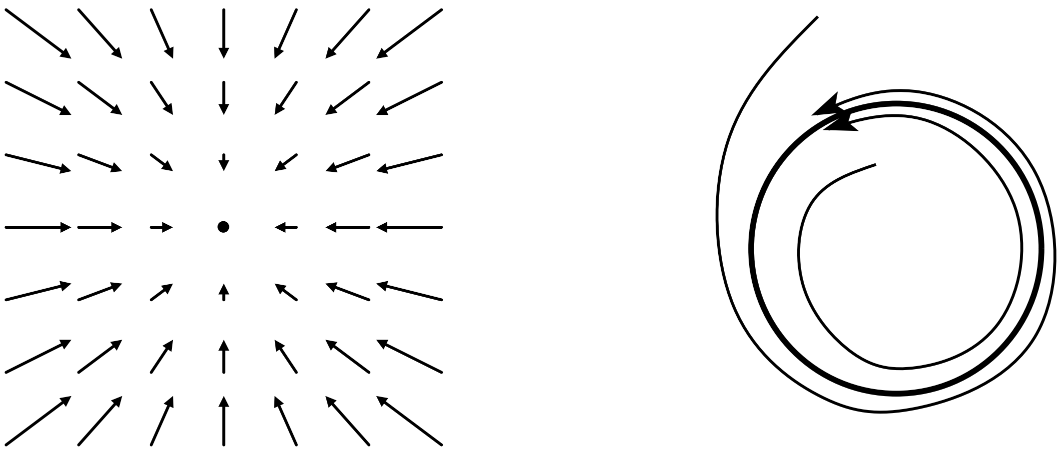


Figure 14.1: A fixed point attractor *left*, and a limit cycle *right*.

Such a system will eventually settle into a state that is a minimum of the energy function.[2]

Example 14.2.1 Friction will eventually slow a swinging pendulum until it settles into the state where the pendulum hangs straight downward. In this state there is no motion energy and the potential energy of gravity is lowest. This state is an attracting state for the system.

In low dimensional systems, we can plot the energy as a function of the state (figure 14.2). From this geometric point of view, the energy function is viewed as an **energy landscape**, in which trajectories of the system "flow downhill." In the example shown, there are three attractors each represented by the lowest energy state at the bottom of a “valley.” The set of all states that flow toward a particular attractor is known as the **basin of attraction** for that attractor. Note that only one of these gives the lowest energy state of the entire system. The other two attractors are **local minima**, *i.e.* they are minima for the energy function in a small neighborhood of the attracting state, but not over the whole state space.

14.3 Hopfield Networks

To make use of an energy landscape to store memories within a network, we must have some rule for structuring a network so that the stored memories lay at local minima of some energy function. In 1982, John Hopfield combined a generalized form of the Hebb rule with a simple activation dynamics, and constructed an energy function in which the stored memory states were indeed local minima of the energy function. This paper was not only important for clarifying how Hebbian learning can lead to attractor memories, it also strengthened the bridge between memory networks and certain branches of statistical mechanics. This opened the field of neural computation to a number of physicists who began to apply sophisticated statistical techniques to understand the behavior of large networks of simplified neurons.

---

[2] To make this statement we have to assume that the energy function is bounded from below, *i.e.* the energy can't grow infinitely negative.

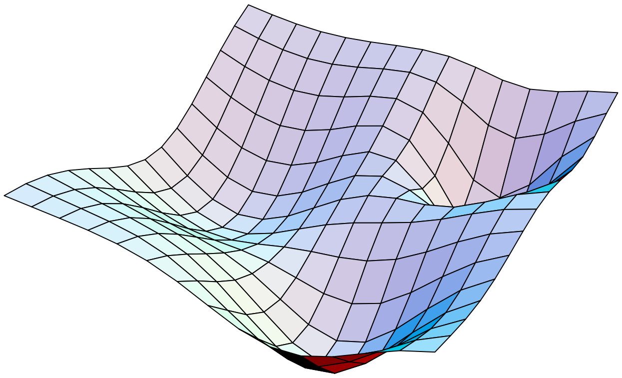


Figure 14.2: An energy landscape with three attractors.

Hopfield's original paper considered binary "McCulloch-Pitts" neurons (see section **??**). However, in 1984 Hopfield demonstrated that the associative memory properties of binary network were quite similar to the continuous time dynamical systems that we have been considering, at least in the limit where the input/output function was a high-gain sigmoid, *i.e.* a smoothed version of the step-like McCulloch-Pitts input/output function (figure 14.3).

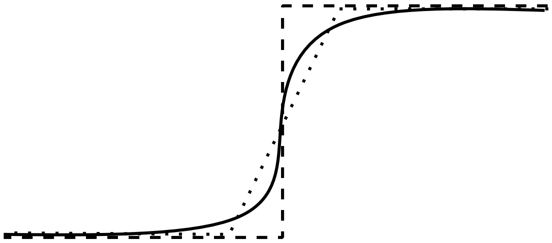


Figure 14.3: Step-like McCulloch-Pitts input/output function (*dashed*), a high-gain sigmoid function (*solid*), and a piecewise linear function (*dotted*).

14.3.1 Constructing the Network

Hopfield networks are built to store L memories, where each memory vector $\mathbf{v}^k$ is a random binary pattern of activity distributed over N neurons, with $N > L$. The simplest version of the network views the binary patterns as strings of 1's and -1's rather than 1's and 0's. Storage and recall are strictly separated in these networks. First, to store the given memories, a matrix of connections $\mathbf{T}$ is constructed according to a correlation-based (Hebbian) outer product rule:

$$\mathbf{T} = \frac{1}{N}\sum_k \mathbf{v}^k(\mathbf{v}^k)^{\mathrm{T}} \tag{14.1}$$

Self-connection strengths $\mathbf{T}_{ii}$ are then set to zero (we will return to this issue below). Note that during the storage phase, activities are assumed to be fixed or **clamped** in the pattern of the memories to be stored, without being influenced by the storage of previous memories. During recall, the activation dynamics follow the usual equations:

$$\tau \dot{u}_i = -u_i + \sum_j \mathbf{T}_{ij} g(u_j) \tag{14.2}$$

g is the sigmoid input/output function. In vector notation this becomes

$$\tau \dot{\mathbf{u}} = -\mathbf{u} + \mathbf{T} g(\mathbf{u}) \tag{14.3}$$

where $g(\mathbf{u})$ is the vector obtained by applying the function g to each element of the vector $\mathbf{u}$. Cued recall works as follows. The "cue" is input to the network in the form of an initial condition $\mathbf{u}(0)$. When the memory network is working properly, the resulting trajectory of output values $g(\mathbf{u}(t))$ flows toward one of the stored memory vectors $\mathbf{v}^k$.

Network Aside. Dynamics in terms of output rather than state variables. MORE.

The most common explanation of how the memory works is simple. Given the current state of network, the output values are given by $\mathbf{r}(t) = g(\mathbf{u}(t))$, and the total input vector

$$\mathbf{T} g(\mathbf{u}(t)) = \sum_k \mathbf{v}^k (\mathbf{v}^k)^{\mathrm{T}} \mathbf{r}(t) = \frac{1}{N} \sum_k (\mathbf{v}^k \cdot \mathbf{r}(t)) \mathbf{v}^k \tag{14.4}$$

In other words, the input vector is given as a linear combination of the memory vectors $\mathbf{v}^k$, where each memory vector is weighted by how well it matches the current state, $\mathbf{v}^k \cdot g(\mathbf{u}(t))$. If the state is closest to the memory vector $\mathbf{v}^1$, then the input will biased most strongly in the direction of $\mathbf{v}^1$. Therefore, the input will drive the activity toward $\mathbf{v}^1$, thereby increasing the match to $\mathbf{v}^1$. This sets up a positive feedback system with the trajectory finally approaching the attracting state $\mathbf{v}^1$. We will re-examine this argument below.

14.4 Three Approaches to Analyzing Hopfield Nets

Hopfield networks have been analyzed from three points of view.

14.4.1 Cohen-Grossberg Energy Function

The first is the energy function point of view presented above. In 1983, Cohen and Grossberg showed that the following was an energy function for the dynamics (14.2):[3]

$$E = -\frac{1}{2} g(\mathbf{u})^{\mathrm{T}} \mathbf{T} g(\mathbf{u}) + \sum_j \int_0^{u_j} ds\ g'(s) s \tag{14.5}$$

---

[3] As is common in Science, Hopfield and the team of Cohen and Grossberg converged on the energy function (14.5) from different points of view. Cohen and Grossberg's results were quite general, providing an energy function for a number of biologically inspired dynamic equations. Equation (14.5) is a special case applied to the dynamics (14.2). Hopfield was specifically interested in associative memory networks derived from a Hebb rule. He published an energy function for the discrete version of the problem in 1982, and published an energy function for the continuous case in 1984.

One can show that the time derivative

$$\begin{aligned} dE/dt &= \sum_i -\left(\mathbf{T}g(\mathbf{u})\right)_i g'(u_i)\dot{u}_i + g'(u_i)u_i\dot{u}_i && (14.6)\\ &= \sum_i -\left(\mathbf{T}g(\mathbf{u}) - \mathbf{u}\right)_i g'(u_i)\dot{u}_i && (14.7)\\ &= \sum_i -g'(u_i)\dot{u}_i^2 \leq 0 && (14.8) \end{aligned}$$

(problem 14.4.1).

> Network Aside. In terms of the output variables $r_i = g(u_i)$ rather than the internal state variables u_i, the energy function becomes
>
> $$E = -\frac{1}{2}\mathbf{r}^{\mathrm{T}}\mathbf{T}\mathbf{r} + \sum_j \int_0^{r_j} ds\; g^{-1}(s) \qquad (14.9)$$

14.4.2 Statistical Mechanics

The most important impact of Hopfield's 1982 paper was it clarified the connection between associative memory networks and the subfield of physics known as statistical mechanics, paving the way for the application of a number of sophisticated tools from physics to the analysis of these networks. Although we will not explore these issues in detail, we will give a brief introduction to the approach.

First, we explore the "high gain" case where $g(u_i) = r_i \approx \pm 1$. Then a stable equilibrium for the dynamics will be found at states where each component of the total input $(\mathbf{Tr})_i$ has the same sign as $\mathbf{r}_i$. This follows immediately from the dynamic equation (14.3), since if $(\mathbf{Tr})_i > 0$ then u_i approaches a positive value and hence r_i remains positive. Suppose that we want to check whether the memory vector $\mathbf{v}^1$ is indeed a stable attractor for the dynamics. From equation (14.4), we have that the total input

$$\begin{aligned} \mathbf{T}\mathbf{v}^1 &= \frac{1}{N}\sum_k (\mathbf{v}^k \cdot \mathbf{r})\mathbf{v}_k && (14.10)\\ &= \mathbf{v}^1 + \frac{1}{N}\sum_{k\neq 1} (\mathbf{v}^k \cdot \mathbf{v}^1)\mathbf{v}_k && (14.11) \end{aligned}$$

The main key to the statistical mechanics approach is that since we are assuming that the memory vectors to be stored were random binary vectors. Therefore, the second term in equation (14.11) can be seen as a "noise term" describing the random interference from other memory vectors. The term $\mathbf{v}^1$ constitutes the "signal." Given this framework, one can then meaningfully speak of the probability that a typical memory vector is stable, where the average is taken over the range of particular memory networks constructed from random memory vectors. Looking at things in more detail we have that

$$\frac{1}{N}\mathbf{v}^k \cdot \mathbf{v}^1 = \frac{1}{N}\sum_j \mathbf{v}_j^k \mathbf{v}_j^1 \qquad (14.12)$$

Since $\mathbf{v}^k$ and $\mathbf{v}^1$ are uncorrelated binary vectors, $\mathbf{v}_j^k\mathbf{v}_j^1$ is a random number taking values 1 or -1. Therefore, $\frac{1}{N}\mathbf{v}^k \cdot \mathbf{v}^1$ is a random number between 1 and -1 with mean value 0 and variance equal to $1/N$. If N is large, this implies that $\frac{1}{N}\mathbf{v}^k \cdot \mathbf{v}^1 \approx 0$ with high probability. This is a special case of the general fact that *random vectors in high dimensional spaces are nearly orthogonal.* Adding the contribution from each of the $L-1$ memory vectors other than $\mathbf{v}^1$ we have that the noise term in equation (14.11) will have mean 0 and variance $(L-1)/N$.

14.4.3 The Brain-State-in-a-Box

Now we look at Hopfield dynamics from the geometric point of view that we have emphasized in the rest of these notes. For this analysis we will use the piecewise linear approximation to the sigmoid function (figure 14.3, *dotted line*). Then the input/output function becomes

$$g(u) = \begin{cases} -1, & u < -1/\tilde{g} \\ \tilde{g}u, & -1/\tilde{g} < u < 1/\tilde{g} \\ 1, & u > 1/\tilde{g} \end{cases} \tag{14.13}$$

Therefore, if all of the neurons are in the linear portion of their input/output functions, the dynamics become linear:

$$\tau\dot{\mathbf{u}} = -\mathbf{u} + \tilde{g}\mathbf{T}\mathbf{u} \tag{14.14}$$

As in all linear dynamics, we look for the eigenvectors of $-\mathbf{I} + \tilde{g}\mathbf{T}$. To begin with, we will assume that the memory vectors used to construct $\mathbf{T}$ are orthogonal. As we saw previously, this is a good approximation as long as the number of memories is small relative to the number of neurons. In this case, each of the memories are eigenvectors, with eigenvalue $\tilde{g} - 1$ (problem 14.4.2). Any vector perpendicular to the subspace spanned by the memory vectors is also an eigenvector, but with eigenvalue -1. Since we are assuming that $\tilde{g}$ is large, the dynamics is exponentially expanding within the memory subspace, and exponentially decaying in directions perpendicular to this subspace (figure 14.4a).

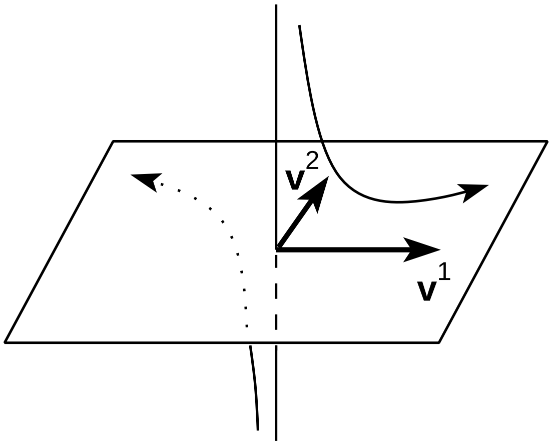


Figure 14.4: Associative memory dynamics for piecewise linear input/output functions before saturation.

As activities grow large, the exponential expansion within the memory subspace is halted when the neurons begin to saturate. If we look at the dynamics of the output values r_i these are prevented from growing beyond the values -1 and 1, *i.e.* the set of allowable output states is the set of all vectors $\mathbf{r}$ with $-1 \leq r_i \leq 1$. Thus the state space can be viewed as a high-dimensional cube (a **hypercube**) in output space, with the states growing exponentially until they reach the faces of this "box." In 19??, James Anderson explored associative memory dynamics from this point of view, calling his model the **brain-state-in-a-box** model.

This model gives a very different perspective on the nature of attractors in these kind of associative memory networks. In particular, within the memory subspace, any linear combination of memory vectors is growing just as fast as the memory vectors themselves. Thus, the argument given in section 14.3.1 is highly misleading. Storing memories using an outer product matrix *in and of itself* does not make the memories into attracting states. It only specifies a memory subspace. The creation of attractors relies crucially on the sigmoid shape of the input/output function. In particular, *saturation of outputs not only serves as a mechanism for bounding the unstable positive feedback created by strong recurrent connectivity, it provides a* **constraint surface** *that greatly influences the attracting states of the network.* From the brain-state-in-a-box point of view, trajectories within the memory subspace expand until they hit the sides of the box, but continue to expand until they reach the corners (figure 14.5b, *left*). From the energy function point of view, attractors are not created by making "dips" in specific locations around memory vectors as suggested by figure 14.2. Rather, the energy function takes the form of a round hump in the memory subspace, with low energy states found in the corners (figure 14.5b, *center*). Note that in higher dimensions the energy function looks like a high-dimensional saddle, with directions perpendicular to the memory subspace represented by the high ends of the saddle and the memory subspace represented by the low ends (figure 14.5b, *right*).

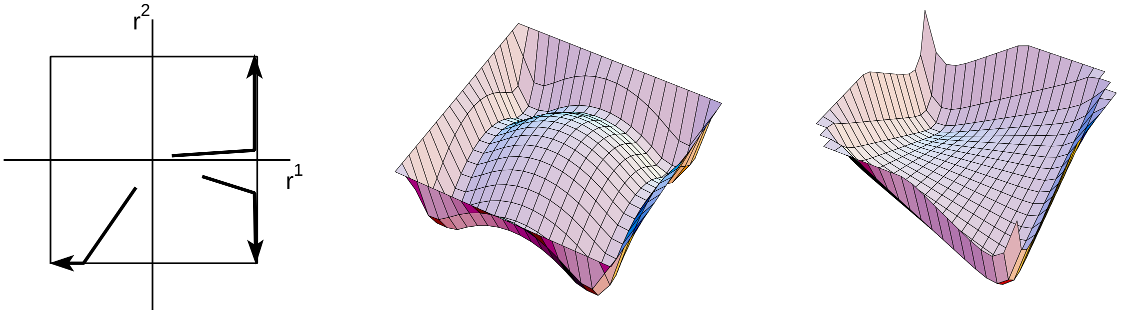


Figure 14.5: Brain-state-in-a-box.

Low dimensional depictions of energy functions and memory subspaces may be misleading. How do we know that states actually end up in the corners of the box? From the energy function point of view, we know that the final states of the network are equilibrium points that are local minima of the energy function. MORE... T_{ii} and gain.

Problems

Problem 14.4.1 Derive equation (14.6). As argued above, this shows that equation (14.5) is indeed an energy function for the dynamics (14.2).

Problem 14.4.2 Show that for $\mathbf{T}$ constructed from the outer product rule using orthogonal memory vectors that each of these vectors is an eigenvector for $-\mathbf{I} + \tilde{g}\mathbf{T}$ with eigenvalue $\tilde{g} - 1$. Show also that any vector perpendicular to the subspace spanned by the memory vectors is also an eigenvector, but with eigenvalue -1.

14.5 Spurious Attractors

So far we have focused on conditions ensuring that the stored memory vectors are stable fixed points for the dynamics. Even if this is the case, the network does not necessarily perform perfectly. In particular, we have not addressed the very real possibility that *other* states besides the memory states may also be attractors. Such attractors are sometimes called **spurious attractors** or **spurious memories**.

The most common type of spurious memories are those constructed from combinations of odd numbers of memories.[4] For example, consider a vector v^{mix} formed by first adding the entries of three memory vectors $\mathbf{v}^1$, $\mathbf{v}^2$, and $\mathbf{v}^3$. This will yield a vector whose entries are -3, -1, 1, or 3. v^{mix} is then formed by setting negative entries to -1 and positive entries to 1. On average, an entry of v^{mix} will match the entry for any of it's component vectors 3/4 of the time. To see this, suppose $\mathbf{v}^1_j = 1$. Then $v^{mix}_j = -1$ only if both $\mathbf{v}^2_j = -1$ and $\mathbf{v}^3_j = -1$, something that should happen 1/4 of the time. If we look at the total input when the network is in the state v^{mix}, we have that

$$\begin{aligned} \mathbf{T}v^{mix} &= \frac{1}{N}\left((v^{mix}\cdot\mathbf{v}^1)\mathbf{v}^1 + (v^{mix}\cdot\mathbf{v}^2)\mathbf{v}^2 + (v^{mix}\cdot\mathbf{v}^3)\mathbf{v}^3 + \sum_{k>3}(\mathbf{v}^k\cdot\mathbf{v}^1)\mathbf{v}_k\right) && (14.15) \\ &\approx \frac{3}{4}(\mathbf{v}^1+\mathbf{v}^2+\mathbf{v}^3) + \frac{1}{N}\sum_{k>3}(\mathbf{v}^k\cdot\mathbf{v}^1)\mathbf{v}_k && (14.16) \end{aligned}$$

If we consider large networks, then the last "noise" term is small, the sign of $\mathbf{T}v^{mix}$ will be determined by $\mathbf{v}^1+\mathbf{v}^2+\mathbf{v}^3$, and so v^{mix} will be a stable fixed point. However, because of the factor of 3/4, v^{mix} is not as stable as $\mathbf{v}^1$, $\mathbf{v}^2$, or $\mathbf{v}^3$. At the entries where all three memories do not agree, it would take smaller values of the noise term to destabilize v^{mix}. From the energy function perspective, v^{mix} is a local minimum of the energy function, but the energy is not as low as for the memory states $\mathbf{v}^1$, $\mathbf{v}^2$, and $\mathbf{v}^3$. This is shown in figure 14.6 using a dimensional general energy schematic as well as a two dimensional "corner" schematic. In both figures the mixture state is represented by the state with higher energy.

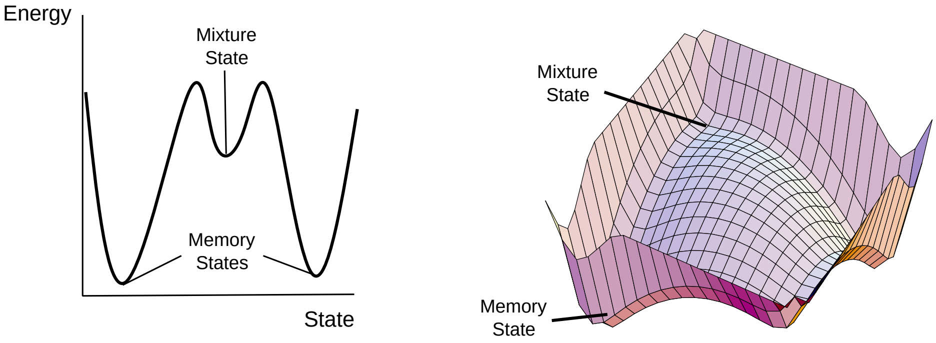


Figure 14.6: Schematic pictures of mixture states as local minima of an energy function, but at higher energy level than the memory states.

One can think about spurious from the brain-state-in-a-box point of view as well, but this takes some imagination. The key is to find some way of picturing how a low dimensional plane is bounded by a high dimensional cube. One can think of the high dimensional cube as a "pointy ball" that bounds the overall size of the activity but allows some vectors – the "points" or "corners"

---

[4] When many memories are stored, spurious memories that are not simply linear combinations of low numbers of memories can also be stable. These are the so-called spin-glass states.

– to be longer than others. As before there are two components to the dynamics, one that leads to outward expansion within the memory subspace and another that leads to decay toward the memory subspace. For points on the ball that are near memory subspace, the memory subspace expansion component will dominate and these points can be stable. Because v^{mix} is nearly a linear combination of the memory vectors $\mathbf{v}^1$, $\mathbf{v}^2$, and $\mathbf{v}^3$, it lies near enough to the memory subspace to be stable.

14.5.1 Lowering the Gain

So far we have discussed the Hopfield network for units with a high gain sigmoid, *i.e.* where the sigmoid is a good approximation to the binary input/output function. What happens if we lower the gain? From the statistical mechanics perspective, this has been shown to be similar to the effects of raising the "temperature" of a number of interacting particles. Although still tending toward the lowest energy state, particles have some probability to jump up from a lower to higher energy state. From the perspective of figure 14.6, high energy mixture state will become relatively unstable since particles can jump out of relatively shallow minima, but will remain in lower energy states. From the box perspective, lowering the gain "smoothes" the box constraint. Figure 14.7 shows how reducing the gain affects the energy function depicted on the right in figure 14.6. The overall energy function is much more "bowl-like," and the corners corresponding to the mixture states are no longer stable fixed-points for the dynamics.

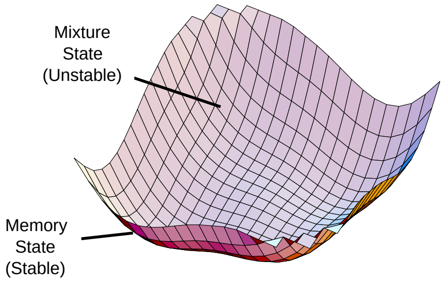


Figure 14.7: Schematic picture of the effect of lowering the gain of the sigmoid on the energy function.

14.6 Biological Realism of Attractor Networks

The Hopfield model demonstrates that attracting states can arise from simple correlation-based learning rules applied to distributed patterns of activity in a recurrent network. However, the model suffers from a number of assumptions that are highly non-biological. Many of these can be addressed with only minor changes to the Hopfield framework. Others lead to networks with attracting states whose stability depends different dynamical mechanisms.

The most common trick that we will use relies on the fact that in large networks, the average activity across all neurons is very close to the average of the distribution used to pick each element in the random memory vectors. (For random binary strings of ±1s, the average value is zero.) This fact will allow us to effectively alter this mean value by adding or subtracting a constant value. Because this value is constant, it can be contributed by non-specific biological mechanisms that affect all neurons equally. To make things simple, we will assume that memory vectors are chosen so that for the various functionally important parameters, the value of that parameter is exactly equal to the mean value of the distribution of parameter values that would result if the memory vectors were chosen truly at random. For example, we will assume that the memory vectors are chosen so that the mean value of the elements in *each* memory vector is exactly equal to the mean of the distribution used to choose each element. For random binary ±1 vectors, this assumption means that we assume an exactly equal number of 1s and -1s. For large networks, the mean over each memory vector will be very close to 0. To the degree that these differ, the more realistic networks outlined below will behave quite very similar to, but not exactly like, the original Hopfield model.

14.7 Postive and Negative Activity Values

The most glaring problem with the original Hopfield network is the use of positive and negative output values. Negative outputs obviously cannot be interpreted as corresponding to firing rate. This problem is easily addressed by simply shifting the range of the input/output function so that it ranges between 0 and 1, and then re-expressing the original dynamics and learning rule to compensate for this change of coordinates.

In actuality, we can use essentially the same learning rule. For binary memory vectors constructed from random choices of -1 and +1, the mean activity level is 0. Therefore, the previous *correlation* learning rule

$$\mathbf{T} = \frac{1}{N}\sum_k \mathbf{v}^k(\mathbf{v}^k)^{\mathrm{T}} \tag{14.17}$$

is equivalent to the *covariance* learning rule

$$\mathbf{T} = \frac{1}{N}\sum_k (\mathbf{v}^k - \mu)(\mathbf{v}^k - \mu)^{\mathrm{T}} \tag{14.18}$$

where μ is the average activity value. Using the covariance learning rule for random binary vectors of 0s and 1s, $\mu = 1/2$ and the resulting matrix is the same as for the corresponding case using ±1, up to a factor of 1/2. To compensate for this factor of 1/2, we will use

$$\mathbf{T} = \frac{2}{N}\sum_k \sum_k (\mathbf{v}^k - \mu)(\mathbf{v}^k - \mu)^{\mathrm{T}} \tag{14.19}$$

Note that for equations (14.17) and (14.18) to be truly equivalent, the mean activity level for each neuron averaged over the memory vectors has to be zero. This will be approximately true if a large number of memory vectors have been stored.

Now we simply use the same dynamics as before. One thing to note is that by shifting the input/output function so that firing rates are positive, we are assuming that $g(0) = 1/2$. Thus, with no input from the network, each neuron will fire at one half its maximal firing rate, *i.e.* this network requires spontaneously active neurons. To achieve the same qualitative dynamics, we need the memory subspace to pass through the center of the output space hypercube. It is easy to check (problem **??**) that the center of the output space is indeed an equilibrium of the dynamics, and hence is included in the memory subspace picture that applies to the linear range of the dynamics (before saturation effects become significant).

14.8 Positive and Negative Connection Strengths

Another way in which the Hopfield network differs from biology is that single neurons in the network can have both excitatory and inhibitory effects on their postsynaptic targets. This apparent conflict may be one of interpretation. If we interpret the units in the Hopfield network not as single neurons, but as groups of neurons functioning as a single unit (*e.g.* a cortical column). Then an inhibitory weight can be interpreted as a strong connection from the excitatory neurons in one population to the inhibitory neurons in the other (or vice versa). While this interpretation can explain the existence of positive and negative weights, it also carries with it additional assumptions. In particular, the rule for plasticity between excitatory and inhibitory neurons must be such that the net connection strengths between populations follow a correlation-based rule.

An alternative to having separate inhibitory neurons coupled to each group of excitatory cells is to posit the existence of a single population of inhibitory neurons that both receives from and projects to all excitatory neurons with equal strength. If this inhibitory population displays a linear response, it's activity will be proportional to the total excitatory activity. Letting h denote this inhibitory activity, we have

$$h = \sum_j W_{he} g(u_j) \tag{14.20}$$

If the inhibitory population is projects back to each excitatory cell with strength W_{eh},

$$\begin{aligned} \tau \dot{u}_i &= -u_i + \sum_j T_{ij} g(u_j) - W_{eh} \sum_j W_{he} g(u_j) && (14.21) \\ &= -u_i + \sum_j (T_{ij} - W_{eh} W_{he}) g(u_j) && (14.22) \end{aligned}$$

Using this model, a negative connection strength between excitatory neurons can be interpreted as indicating that the excitatory connection is weaker than the mutual inhibitory influence mediated by the inhibitory population. This interpretation has the benefit of not requiring new assumptions on the learning rule. The Hebb rule can be seen as increasing or decreasing a default positive excitation between neurons depending on whether the neurons are correlated or anti-correlated over the memory vectors. The feedback inhibition is assumed to counteract this positive excitation leaving the original Hopfield proscription to determine the net effective connectivity.

14.9 High Firing Rates and Synaptic Saturation

We've seen that Hopfield networks act as brain-states-in-a-box and that attractors are formed as trajectories move into the corners of the box. Memory vectors are binary, and the corners of the box are determined by neurons running up against the physiological bounds of low (near zero) and high (near maximal) firing rates. However, physiological recordings from neurons in the circuits that have been hypothesized to form attractors reveal that these neurons tend to fire at relatively modest firing rates, far below the rates that the cells can be driven to by artificial stimulation or that are displayed by similar cells performing different computations.

One possibility that may explain the apparent discrepancy between the model and the data is that the necessary saturation may be found at the synapse level rather than in the neuronal input/output function. A number of mechanisms may cause a reduction in effective synaptic strength at synapses where the presynaptic neuron has elevated activity levels for extended periods of time. This synaptic depression may be caused by the depletion of releasable neurotransmitter in the presynaptic terminal or by saturation or densensitization of postsynaptic receptors. The postsynaptic effects may be particulary prominent for slow synaptic currents, such as NMDA currents. One simple way to incorporate these effects is to assume that the net synaptic current contributed by neuron j to neuron i is equal to $T_{ij}f(r_j) = T_{ij}f(g(u_j))$, where $r_j = g(u_j)$ is the firing rate of neuron j. This raises the possibility that the transfer of presynaptic activation to postsynaptic current could saturate well-below the saturation level of the input/output function alone. This synaptic saturation will bound the total synaptic current received by neurons in the network, and hence could act to bound their firing rates at physiologically low values.

These notes have focused on models where synaptic integration is modelled as a static linear operation. While this simplification is useful for thinking about processing within neural circuits, one must always remember that synapses and dendrites are complex and dynamic devices, and this complexity is likely to have a significant impact on the nature of computation within neural circuits.

14.10 Low Gain Networks

Problems

Problem 14.10.1 Show that the vector of equal activities being a stable eigenvector for the connection matrix is equivalent to the condition that the sum of connection strengths onto each neuron is a constant that is less than 1.

Chapter 15

Oscillatory Dynamics

15.1 Linear Oscillators

The dynamical systems examined so far all used symmetric matrices. There are two basic reasons for studying such systems. First, symmetric matrices arise naturally in many contexts. For example, correlation-based learning relies on correlation matrices, and these are symmetric. Also, distance-based connectivity at the population level is a reasonable assumption in many circuits. The second reason for studying symmetric matrices is that these matrices are special: every symmetric matrix has a complete orthonormal eigenbasis. This makes analysis easy.

One case where the matrix governing the dynamics is fundamentally asymmetric is networks of interconnection excitatory and inhibitory neurons. By definition, the connection in one direction is positive and the reverse connection is negative. The first half of this chapter will be devoted to studying the linear version of the simplest such network, one that has one excitatory and one inhibitory population (see figure 15.1). The dynamics governing this network is given by

$$\begin{aligned} \tau_e \dot{e} &= -e + T_{ee} e - T_{eh} h + q_e & (15.1) \\ \tau_h \dot{h} &= -h + T_{he} e - T_{hh} h + q_h & (15.2) \end{aligned}$$

We have used h to represent the activity in the inhibitory population since the use of i here could be confused with one of this variable's other uses. Note that T_{xx} is always positive; the negative weights are written $-T_{eh}$ and $-T_{hh}$. To write this equation in matrix form, we must remember that we have included different time constants for the excitatory and inhibitory populations. Therefore, if we write the activity vector $\mathbf{u} = [e, h]^{\mathrm{T}}$, we divide by the time constants τ_e and τ_h and write the dynamics as

$$\dot{\mathbf{u}} = \mathbf{D}(-\mathbf{I} + \mathbf{T})\mathbf{u} + \mathbf{D}\mathbf{q} \qquad (15.3)$$

where division by the time constants is accomplished by multiplication by the 2×2 diagonal matrix $\mathbf{D}$ with entries $1/\tau_e$ and $1/\tau_h$ (problem 15.1.1). We will begin by examining the conditions under which the fixed point in equations (15.1) and (15.2) is a globally attracting (stable) equilibrium. We will go on to examine two different ways in which this point can lose stability, and how these can give rise to oscillatory dynamics.

Problems

Problem 15.1.1 (E) Show that multiplication by the matrix $\mathbf{D}$ divides the entries in the top row by τ_e and divides the bottom row by τ_h.

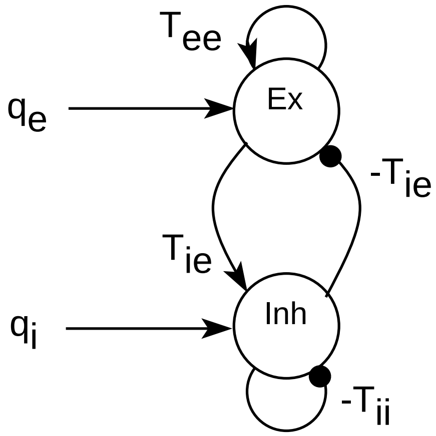


Figure 15.1: Simple excitatory-inhibitory network.

15.2 Complex Eigenvalues

As before, to solve this equation we will look for the eigenvalues and eigenvectors of the matrix governing the dynamics. For 2×2 matrices $\mathbf{A}$, it can be shown that the eigenvalues λ are solutions to the following quadratic equation:

$$\lambda^2 - (A_{11} + A_{22})\lambda + (A_{11}A_{22} - A_{12}A_{21}) \tag{15.4}$$

From the quadratic formula we have that

$$\lambda = \frac{1}{2}\left((A_{11} + A_{22}) \pm \sqrt{(A_{11} + A_{22})^2 - 4(A_{11}A_{22} - A_{12}A_{21})}\right) \tag{15.5}$$

As long as the term under the square root sign is positive, the equation yields two real-valued eigenvalues. However, if the sign under the square root sign is negative, then the eigenvalues are a pair of complex numbers.

> Mathematical Aside. Very briefly, **complex numbers** are numbers that can be written $a + ib$ where a and b are scalars and $i = \sqrt{-1}$. a is known as the **real part** and b the **imaginary part**. Such numbers are often represented as points on the **complex plane**, where the horizontal axis is the **real axis**, and the vertical axis is the **imaginary axis**. The most important fact about complex numbers that we will use is Euler's equation:
>
> $$a + bi = re^{i\theta} \tag{15.6}$$
>
> where r and θ are the radius and angle of the polar coordinate representation the point $a + bi$ (see figure 15.2a and problem **??**). A complex vector is just a vector whose elements are complex numbers. Any complex vector $\mathbf{v}$ can be written as a sum $\mathbf{v} = \mathbf{v}_r + i\mathbf{v}_i$ where $\mathbf{v}_r$ and $\mathbf{v}_i$ are real vectors.

Introducing a couple of fundamental quantities about matrices will give a better understanding of the equation (15.5).

Definition 19 *For a square matrix* $\mathbf{A}$*, the sum of the diagonal elements of* $\mathbf{A}$ *is called the* **trace** *of* $\mathbf{A}$*, Trace(*$\mathbf{A}$*). It can be shown that Trace(*$\mathbf{A}$*) is equal to the sum of the eigenvalues of* $\mathbf{A}$*.*

Definition 20 *For a square matrix* $\mathbf{A}$*, the product of the eigenvalues is called the* **determinant** *of* $\mathbf{A}$*, Det(*$\mathbf{A}$*). Note that the determinant can be calculated directly as a (fairly complex) function of the entries of* $\mathbf{A}$*, without first finding the eigenvalues.*

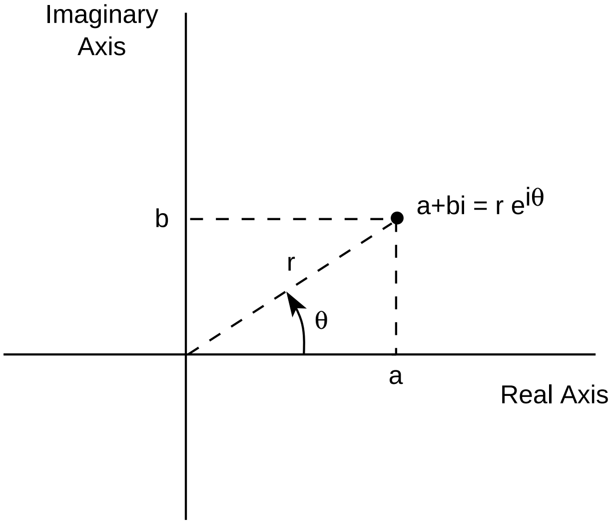


Figure 15.2: The complex plane.

> Mathematical Aside. Since Trace($\mathbf{A}$) is equal to the sum of the eigenvalues of $\mathbf{A}$, this implies that the off-diagonal elements have no effect on the sum of the eigenvalues. Furthermore, since the eigenvalues of a matrix stay the same when the matrix is expressed in different coordinates, this says that the sum of the diagonal elements stays is invariant to changes in coordinates.

Using these definitions, equation (15.5) can be rewritten as

$$\lambda = \frac{1}{2}\left(\text{Trace}(\mathbf{A}) \pm \sqrt{\text{Trace}(\mathbf{A})^2 - 4\text{Det}(\mathbf{A})}\right) \tag{15.7}$$

We proceed as before, finding the corresponding eigenvectors and writing the solution as a linear sum of exponential growth or decay in separate components. Suppose for the moment that we have found a complex eigenvalue and a complex eigenvector, *i.e.*

$$\dot{\mathbf{v}} = \mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{15.8}$$

where $\mathbf{v} = \mathbf{v}_r + i\mathbf{v}_i$ and $\lambda = \lambda_r + i\lambda_i$ are complex. We can proceed formally and write down the exponential solution as before,

$$\mathbf{u}(t) = \mathbf{u}(0)e^{\lambda t} \tag{15.9}$$

where the initial condition $\mathbf{u}(0)$ can be obtained from $\mathbf{v}$ via multiplication by a (potentially complex) scalar α, *i.e.* $\mathbf{u}(0) = \alpha\mathbf{v}$. Our goal now is to use Euler's formula to "unwrap" equation (15.9) and figure out how to think about it back in the more comfortable space of real vectors. The key idea is to view the two dimensional plane spanned by the real and imaginary parts of the eigenvector $\mathbf{v}$ as the "eigendirection" for $\mathbf{v}$. However, we need one more concept regarding complex numbers before we can fully exploit this intuition.

Definition 21 *The* **complex conjugate**, *z^*, of the complex number (or vector) $z = a + ib$ is obtained by changing the sign of the imaginary part of z, i.e. $z^* = a - ib$. We note that the real part of z, $a = (z + z^*)/2$; the imaginary part of z, $b = (z - z^*)/2$. It is easy to check that zz^* is a real number and is equal to the squared magnitude of vector z in the complex plane. This is sometimes written $|z| = \sqrt{zz^*}$. It follows from this definition that if w and z are complex numbers that $(wz)^* = w^*z^*$.*

One can show that *if* $\mathbf{v}$ *is an eigenvector with eigenvalue* λ*, then* $\mathbf{v}^*$ *is an eigenvector with eigenvalue* λ^* (problem **??**).

Recall that, in the case of real eigenvalues and vectors, we have a simple exponential solution only when the initial condition is in the same direction as the eigenvector. Since we are viewing the two dimensional plane spanned by the real and imaginary parts of the vector $\mathbf{v} = \mathbf{v}_r + i\mathbf{v}_i$ as the analogue of the eigendirection, we focus on the trajectories $\tilde{\mathbf{v}}_r(t)$ and $\tilde{\mathbf{v}}_i(t)$ starting at the initial points $\tilde{\mathbf{v}}_r(0) = \mathbf{v}_r$ and $\tilde{\mathbf{v}}_i(0) = \mathbf{v}_i$. We begin by solving for $\tilde{\mathbf{v}}_r(t)$. Note that $\mathbf{v}_r = (\mathbf{v} + \mathbf{v}^*)/2$ and that $\mathbf{v}(t) = \mathbf{v}e^{\lambda t}$ and $\mathbf{v}^*(t) = \mathbf{v}^* e^{\lambda^* t}$ are the equations describing trajectories with initial conditions $\mathbf{v}$ and $\mathbf{v}^*$. Therefore

$$\tilde{\mathbf{v}}_r(t) = (e^{\lambda t}\mathbf{v} + e^{\lambda^* t}\mathbf{v}^*)/2 \tag{15.10}$$

Writing out the real and imaginary parts of λ, λ^*, $\mathbf{v}$ and $\mathbf{v}^*$, we have that

$$\tilde{\mathbf{v}}_r(t) = \frac{1}{2}\left(e^{\lambda_r t}e^{i\lambda_i t}(\mathbf{v}_r + i\mathbf{v}_i) + e^{\lambda_r t}e^{-i\lambda_i t}(\mathbf{v}_r - i\mathbf{v}_i)\right) \tag{15.11}$$

$$= \frac{e^{\lambda_r t}}{2}\left((\cos(\lambda_i t) + i\sin(\lambda_i t))(\mathbf{v}_r + i\mathbf{v}_i) + (\cos(-\lambda_i t) + i\sin(-\lambda_i t))(\mathbf{v}_r - i\mathbf{v}_i)\right) \tag{15.12}$$

$$e^{\lambda_r t}\left(\cos(\lambda_i t)\mathbf{v}_r - \sin(\lambda_i t)\mathbf{v}_i\right) \tag{15.13}$$

A similar derivation shows that

$$\tilde{\mathbf{v}}_i(t) = e^{\lambda_r t}\left(\cos(\lambda_i t)\mathbf{v}_i + \sin(\lambda_i t)\mathbf{v}_r\right) \tag{15.14}$$

(problem **??**). From these formulas we discover that the real part λ_r of the complex eigenvalue λ determines an overall exponential growth or decay of the trajectory. If $\lambda_r > 0$, the term $e^{\lambda_r t}$ will grow exponentially large with time. If $\lambda_r < 0$, $e^{\lambda_r t}$ will decrease exponentially to zero. The imaginary part λ_i of λ determines the "angular velocity" of a sinusoidal oscillation between the vectors $\mathbf{v}_r$ and $\mathbf{v}_i$. Combining these two effects we find that *complex eigenvalues lead to spiral trajectories, with trajectories spiraling away from the equilibrium when* $\lambda_r > 0$ *and spiralling in when* $\lambda_r < 0$ (see figure 15.3b). When $\lambda_r = 0$, the trajectory will oscillate in an elliptical orbit of constant amplitude. These statements are true for any trajectory that lies in the "eigenplane" spanned by $\mathbf{v}_r$ and $\mathbf{v}_i$.

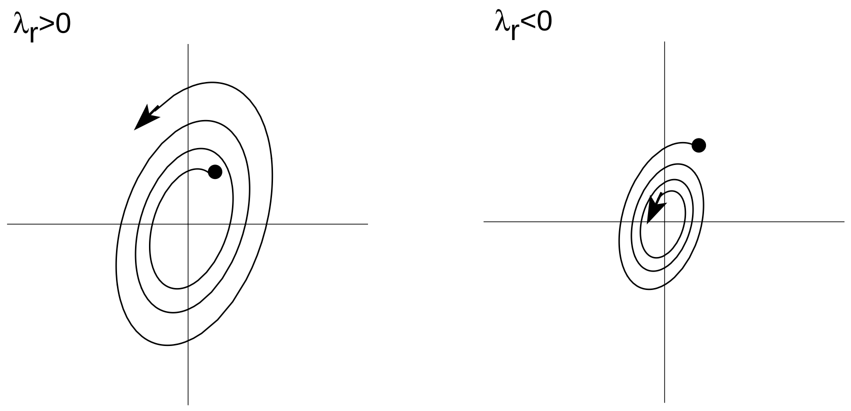


Figure 15.3: Trajectories resulting from linear dynamics with a complex eigenvalue.

Problems

Problem 15.2.1 (E) Write the complex number $a + bi$ in polar coordinates, *i.e.* write the radius and angle of this number in terms of a and b. Write the complex number $re^{i\theta}$ in cartesian coordinates, *i.e.* write the projections along the real and imaginary axis in terms of r and θ.

Problem 15.2.2 (E) Use equation (15.5) to show that the sum of the eigenvalues is equal to Trace($\mathbf{A}$), and the product of the eigenvalues is equal to Det($\mathbf{A}$).

Problem 15.2.3 Derive equation (15.14).

Problem 15.2.4 Show that if $\mathbf{v}$ is an eigenvector of the matrix $\mathbf{A}$ with eigenvalue λ, then $\mathbf{v}^*$ is an eigenvector of $\mathbf{A}$ with eigenvalue λ^*.

Problem 15.2.5 Show that if an eigenvalue of a real matrix is complex (imaginary part is not equal to 0), then the corresponding eigenvector is complex as well.

15.3 Stability of the Excitatory-Inhibitory Network

Before we return to the example of the two-population, excitatory-inhibitory network, we will examine the conditions for stability in a general two-dimensional linear network. First, we consider the case that both eigenvalues are real. This will occur whenever the term under the square root in equation (15.7) is non-negative, *i.e.*

$$\mathrm{Trace}(\mathbf{A})^2 \geq 4\mathrm{Det}(\mathbf{A}) \tag{15.15}$$

Stability requires an exponential *decay* in both eigendirections, hence both eigenvalues must be negative. Examining equation (15.7), we see that this will occur when $\mathrm{Trace}(\mathbf{A}) < 0$ and $\mathrm{Det}(\mathbf{A}) > 0$. Of course these conditions follow directly from the way we have defined the trace and determinant: if both eigenvalues are to be negative, then their sum (the trace) must be negative and their product (the determinant) must be positive. In the case that we have a pair of complex conjugate eigenvalues ($\mathrm{Trace}(\mathbf{A})^2 < 4\mathrm{Det}(\mathbf{A})$), then stability requires that trajectories spiral *in* toward the equilibrium, *i.e.* the real part of the eigenvalues must be less than 0. This will happen when $\mathrm{Trace}(\mathbf{A}) < 0$. Note that since $\mathrm{Trace}(\mathbf{A})^2 < 4\mathrm{Det}(\mathbf{A})$ requires that $\mathrm{Det}(\mathbf{A}) > 0$, we have shown that *the equilibrium for the linear dynamics defined by the* 2×2 *matrix* $\mathbf{A}$ *will be stable whenever Trace*$(\mathbf{A}) < 0$ *and Det*$(\mathbf{A}) > 0$.

Now we return to the example of the two-population, excitatory-inhibitory network and give a biological interpretation to the stability conditions in this case. Plugging the values from the matrix $\mathbf{D}(-\mathbf{I} + T)$, we have

$$\begin{aligned} \mathrm{Trace}(\mathbf{D}(-\mathbf{I}+T)) &= \frac{-1+T_{ee}}{\tau_e} + \frac{-1-T_{hh}}{\tau_h} < 0 && (15.16) \\ \mathrm{Det}(\mathbf{D}(-\mathbf{I}+T)) &= \frac{1}{\tau_e\tau_h}\left((-1+T_{ee})(-1-T_{hh}) + T_{eh}T_{he}\right) > 0 && (15.17) \end{aligned}$$

Starting with the trace condition (15.16), the first thing to notice is that if the excitatory population is stable by itself ($T_{ee} < 1$ – see section **??**), then entire system will be stable. This is not surprising since one would expect that instability requires some subcomponent to be unstable. (Note that the inhibitory population acting alone is always stable.) One is tempted to characterize the trace stability condition (15.16) as saying that instability requires that the excitatory population be "more unstable" than the inhibitory population is stable. However, this is only true if $\tau_e = \tau_h$. These time constants don't influence the stability condition of either population alone, but do influence overall stability. We will return to this issue below.

Focusing on the determinant condition (15.17), we see that since the time constants are always positive they do not affect this condition. To get an intuitive handle on the condition (15.17) we will rewrite it as

$$T_{ee} - 1 < \frac{T_{eh}T_{eh}}{1 + T_{hh}} \tag{15.18}$$

The left-hand side of this equation represents the overall gain of the excitatory population, which we are assuming to be greater than zero. How are we to interpret the right-hand side? The first thing to notice is that $1/(1 + T_{hh})$ is the effective gain of the inhibitory population to step changes in input to this population. As such, the right-hand side represent the net gain of the inhibitory feedback loop that takes a change in excitatory activity, converts this into input to the inhibitory population by multiplying by excitatory→inhibitory weight T_{he}, converts this input into a change in activity in the inhibitory population by multiplying by $1/(1 + T_{hh})$, and finally converts this into an inhibitory input to the excitatory population by multiplying by the inhibitory→excitatory weight T_{eh}. Therefore, we can interpret equation (15.18) quite simply: stability requires that if the activity in the excitatory population is increased slightly, the extra amount of excitation that would result from this self-excitation is smaller than the extra amount of inhibition that would result from this activity propagating through the inhibitory feedback loop. More simply, *stability requires that the feedback inhibition be stronger than the feedback excitation.*

Now we reexamine the trace condition (15.16). We rewrite this condition as

$$\frac{-1 + T_{ee}}{\tau_e} < \frac{1 + T_{hh}}{\tau_h} \tag{15.19}$$

Again suppose that the excitatory population has moved a slight distance Δe from threshold. Then, if all other inputs were to remain fixed, the equation (15.1) governing the excitatory dynamics says that, in the next small time period Δt, the distance from threshold would increase by a factor $\Delta e \Delta t(-1 + T_{ee})/\tau_e$. In other words, $(-1 + T_{ee})/\tau_e$ determines the rate at which the excitatory population moves away from its equilibrium. Similarly, equation (15.2) implies that the rate at which the inhibitory population returns to its equilibrium is determined by $(1+T_{hh})/\tau_h$. Therefore, the trace condition (15.16) can be interpreted as follows: *stability of the full excitatory-inhibitory network requires that the feedback inhibition be faster than the growth of the feedback excitation.*

Combining the trace and determinant conditions, we see that *in order for feedback inhibition to stabilize an unstable excitatory population, the feedback inhibition must be sufficiently strong and sufficiently fast.* One parameter that is particularly interesting to look at in terms of stability is the self-inhibitory weight in the inhibitory population, T_{hh}. Changes in this parameter have opposite effects on the two stability conditions. Decreasing T_{hh} makes the feedback inhibition stronger, since it reduces inhibitory self-inhibition and thereby increases the effective gain of activity as it passes through the inhibitory population. However, decreases in T_{hh} reduce the effective time constant of the inhibition and hence could slow inhibition to a sufficient degree that stability is lost.

15.4 Oscillations Resulting from Loss of Stability

Almost all oscillators can be described at some as an unstable system feedback compensated for by a relatively slow or weak systems of negative feedback. While this situation describes the linear excitatory-inhibitory network, we only had true oscillations in the very special case that the real part of a pair of complex conjugate eigenvalues was exactly 0. More commonly, the dynamics with complex eigenvalues spiraled in or spiraled away from a single equilibrium. To get more robust oscillations we will consider a system where the excitatory inhibitory populations have sigmoid-shaped input/output functions:

$$\tau_e \dot{e} = -e + T_{ee} g_e(e) - T_{eh} g_h(h) + q_e \tag{15.20}$$
$$\tau_h \dot{h} = -h + T_{he} g_e(e) - T_{hh} g_h(h) + q_h \tag{15.21}$$

The easiest way to see how an oscillatory system can result from the loss of stability in a feedback system is to re-examine the bifurcation diagram for a population of nonlinear self-exciting neurons (figure 15.4, *left*; went over in class). Since we want this population to be unstable, we assume that the maximum slope of the sigmoid input/output function, times the self-connection strength T_{ee} is greater than one. In this case, there is a range of inputs over which the excitatory population is bi-stable, *i.e.* the single equilibrium near the half-maximal point of activity is unstable (*dashed line*), and this separates two stable equilibria representing high and low activity levels (*solid lines*). If the external input is sufficiently positive, there is only one, high-activity equilibrium; if the external input is sufficiently negative, there is only one, low-activity equilibrium.

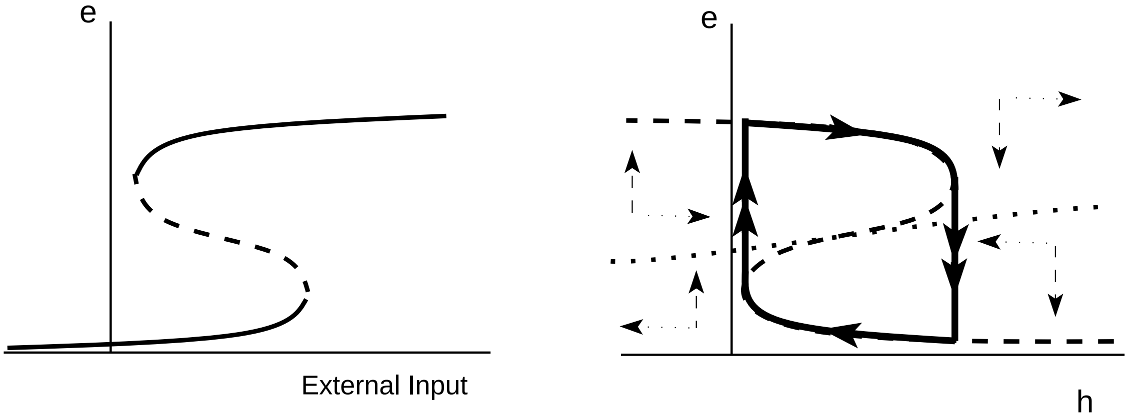


Figure 15.4: Bifurcation diagram for unstable excitatory population (*left*). Phase-plane analysis of oscillatory activity in the related system with slow inhibition.

We can use this bifurcation diagram to understand oscillatory dynamics if we now add *very slow* feedback inhibition, *i.e.* we consider equations (15.20) and (15.20) with τ_h very large. Choosing τ_h to be very large allows us to use the common trick of separating time scales. We have used this trick before in separating activation dynamics from the dynamics of synaptic plasticity. Here the separation of time scales allows us to assume that on the time scale of the excitatory dynamics, the level of inhibition is nearly constant. From the perspective of the inhibitory population, the dynamics of the excitation is so fast, that we need only to consider the equilibrium value for these dynamics.

Under the slow inhibition assumption, we can use the bifurcation diagram for the excitatory population (figure 15.4, *left*), do perform a **phase plane analysis** for the full system (figure 15.4, *right*).

> Mathematical Aside. Phase plane analysis is a mathematical technique that focuses on the geometry of fixed points in low dimensional dynamical systems. Often the phase plane depiction can be used as a way of visualizing a vector field without having to display all the arrows; it is commonly applied to two-dimensional systems. The key element in using the phase plane technique is the plotting of the **null-clines** of the system. A null-cline is simply a set of points where one of the component variables in the system has zero derivative. In a two dimensional system, there will be two null-clines, one for each variable. The null-clines are generally used in two ways. Most importantly, the intersection of the null-clines represents the points where both variables have zero derivative, *i.e.* these intersections represent the equilibrium points of the system. The second way in which null-clines are used are to picture the overall flow of the dynamics. Since the null-cline represents the points where the derivative is zero, for locations on one side of a null-cline, the corresponding variable will be increasing; on one side of the null-cline, the variable is decreasing.

In figure 15.4 (*right*), the dashed line represents the null-cline of the excitatory population. This looks like the mirror image of the bifurcation diagram depicted on the left. This is because both the bifurcation diagram for the excitatory population and the excitatory null-cline for the whole system focuses on the points where $\dot{e} = 0$. The mirroring is due to the fact that in the bifurcation diagram the horizontal axis represents total input, and in the phase plane it represents inhibitory activity. While the structure of the two diagrams is very similar, they represent conceptually different things. In the bifurcation diagram, the external input (including the inhibitory input) was considered as a fixed *parameter*. Each slice in the vertical direction represents a different one-dimensional dynamical system. In the phase plane picture, the level of inhibition is a dynamic *variable*, and the figure represents a single two-dimensional dynamical system. However, because we are assuming that the inhibition changes slowly, the excitatory dynamics acts *as if* the inhibition we a constant parameter, and intuition gained from the bifurcation picture easily transfers to the dynamic case. The inhibitory null-cline, *i.e.* the points where $\dot{h} = 0$ is represented by the dotted line.

To get a feeling for the flow of the dynamics, we need to know not only where the points where $\dot{e}$ or $\dot{h}$ are equal to zero, but also what this indicates about the *sign* of the derivative (whether the derivatives are positive or negative). Above and to the right of the dashed line, the activity in the excitatory population is decreasing ($\dot{e} < 0$). Since e is decreasing, the vector field at these points has a downward component, represented by the downward-pointing dashed arrows. These are points where the inhibition and/or the self-excitation would drive the excitation to a lower level than the current activity. Below and to the left of the solid line, the inhibition and/or the self-excitation would drive the excitation upward, *i.e.* $\dot{e} > 0$. This behavior is represented by upward-pointing dashed arrows. Turning our attention to the inhibitory population, above the inhibitory null-cline the excitatory population is firing vigorously and is driving the inhibitory population to increase its activity level (rightward-point dotted arrows). Below the inhibitory null-cline the excitatory population is relatively inactive and the inhibition is decreasing (leftward-point dotted arrows).

Finally we can use these facts to put the oscillation back together. Let's begin at the "top edge" of the oscillation. Activity in the excitatory population has just increased to a high level. This high level of activity begins to excite the inhibitory population and the state of the system slowly drifts to the right. The accumulating inhibition slightly depresses the activity level until at some point the feedback excitation is no longer self-sustaining: the loss of excitatory activity leads to decreased excitation and a further decrease in excitatory activity. This results in a rapid transition from the upper to the lower "arm" of the excitatory null-cline. But now since the excitatory population is relatively inactive, the inhibition begins to decay, and the state slowly drifts to the left. The decaying inhibition leads to slowly rising levels of activity until at some point the positive feedback is too great and the system becomes unstable again. The positive feedback in excitatory population leads to a rapid rise in excitatory activity, and we are back to where we started from.

Note that this same qualitative picture can be used to describe a whole host of oscillatory phenomena. ranging from the scale of single cells and below to perceptual phenomena and above. We describe a few below.

Biological Aside. Bursting neuron.

Biological Aside. Spike generation.

Biological Aside. Perceptual rivalry.

Biological Aside. Half-center oscillators for locomotion.

Chapter 16

Mathematical Preliminaries

In general, these notes assume don't assume math knowledge beyond some basic high school math and calculus. That doesn't mean that all these ideas will be fresh in the reader's mind. In this chapter, I review some very basic definitions to stir some of the memories. To get a general orientation, the math involved in this course can be divided into three main topics: calculus, probability theory, and linear algebra. (I also throw in a selection of other useful facts.) The main notions of linear algebra are presented in the main body of these notes. This chapter will refresh some topics in calculus and probability theory. Many readers can easily skip (or skim) this chapter (although I do suggest reading the next section).

16.1 Continuous and Discrete Math

Many of the computational issues encountered in these notes can be approached using either continuous or discrete mathematics. As an example, suppose one is interested in modelling interactions among a number of neurons in the visual cortex that are tuned to the direction of motion of images across the retina. To keep things simple, one could examine some number of neurons, say N, and give each one an **index** or label from the set $\{1, 2, \ldots, i, \ldots, N-1, N\}$. Suppose the optimal directions in this population are spread equally over all directions, *e.g.* we could set the preferred direction of the ith neuron to be equal to $360i/N$. Taking $N = 18$, neurons will be spaced every 20^o. Note that we have to (arbitrarily) assign angles to directions in the world (say 0 degrees represents motion to the right and 90 degrees represents upward motion - figure 16.1A). Now suppose that a stimulus comes across the retina moving in the direction 123^o. Then we can plot the response of our 18 neurons arranged according to their preferred direction θ (figure 16.1B, *dots*). The total response is simply a list of 18 numbers. Such a list, $\mathbf{r} = \{r_1, r_2, \ldots, r_N\}$, is called a **vector** (introduced in detail in chapter **??** below). i is called the **index** of the ith element in the vector $\mathbf{r}$.

This is the discrete description of this response. Now imagine that we add more and more neurons, so that the spacing between the preferred directions gets closer and closer. The limit of an infinite number of neurons is called the **continuum limit**. The response can no longer be characterized by a vector, but now is more properly viewed as a function $r(\theta)$ that assigns a response strength to every preferred direction θ (figure 16.1B, *continuous line*). Of course a vector can be viewed as a function as well - the function simply takes the index i and maps it onto the value r_i, *i.e.* $\mathbf{r}(i) = \mathbf{r}_i$.

Obviously, the discrete and continuous formulations are closely related, at least when there are a lot of neurons. The math for the two situations is also closely related, although things can sometimes get tricky when going to the continuum limit. We won't worry about such mathematical fine points, and will freely adopt the discrete or continuous perspectives, whichever seems most natural for the problem at hand. We will rely on an intuitive understanding of how the two pictures relate as the number of neurons grows large.

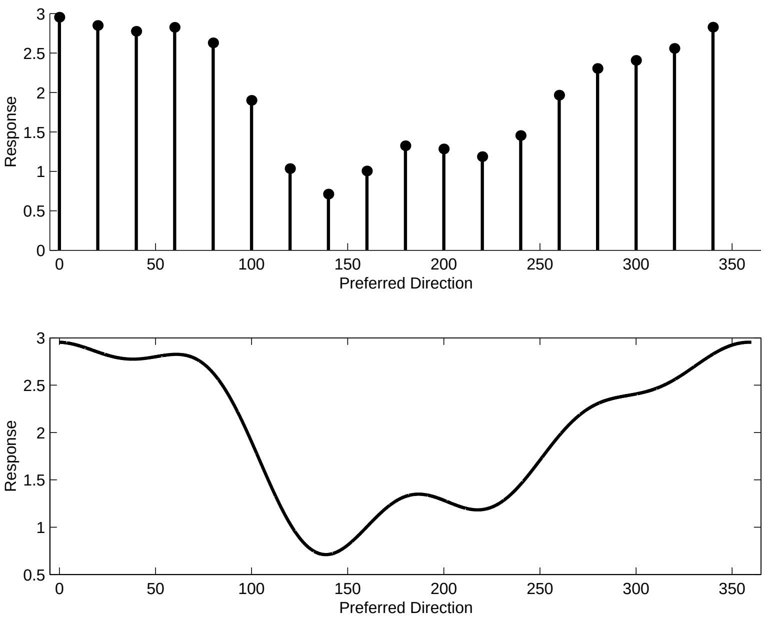


Figure 16.1: Discrete and Continuous Models of Direction Tuning

16.2 Keeping Track of Units

During the course, we will be manipulating a number of formulas. The variables that are involved in these formulas are meant correspond to biological variables such as firing rate, current, etc. In parsing these formulas, it can be quite useful to keep track of the units of the underlying parameters. “Keeping track” of units means including the units into the equations, and treating them much like the underlying variables.

The simplest version comes when one changes units. For example, suppose you know the speed of sound in air is 600? miles/hour. However, you are interested in expressing this speed in meters per second. You know that there are 1.6 kilometers in a mile, 1000 meters in a kilometer, 60 minutes in an hour and 60 seconds in a minute. First we will express the speed in units of meters/hour:

$$\frac{600 miles}{hour} = \frac{600 miles}{hour}\frac{1.6 kilometers}{1 mile}\frac{1000 meters}{1 kilometer} = \frac{xxx meters}{hour} \tag{16.1}$$

Then we change the temporal units:

$$\frac{xxx meters}{hour} = \frac{xxx meters}{hour}\frac{1 hour}{60 minutes}\frac{1 minute}{60 second} = \frac{xxx meters}{seconds} \tag{16.2}$$

Sometimes it is useful to use negative exponents for units in the denominator, $i.e.\ xxxmeters/seconds = xxxmeterssecond^{-1}$.

There are a number of quantities that don’t have units in a strict sense. These arise in two situations. First, there is the case where one is simply counting the number of items or events. An example of this might be the number of spikes recorded from a single neuron during the presentation of a given stimulus. If one divides by the duration of the stimulus this number can be converted in a rate. For example, if one records 10 spikes during a 2 second stimulus, the rate would be

$$\frac{10 spikes}{2 sec} = 5 spikes/sec = 5 Hz \tag{16.3}$$

The units Herz (Hz) denotes events per second, *i.e.* $Hz = 1/sec$. Note that including the units "spikes" is optional, since this just tells us which kind of events are being counted. It is common to include these optional units when using the $1/sec$ notation, but not when using Hz.

The second situation where one encounters quantities without units is when quantities are derived as a ratio of two similar quantities. One of the best example of this kind of unitless quantity is a percentage. For example, having 75% humidity means that the amount of moisture in a given volume, is a given fraction of the total moisture that would saturate that air. Even though percentages are written with a % sign, percentages can be converted to a fraction by dividing by 100, *i.e.* $75\% = .75$. So if there is 34 milligrams (mg) of water suspended in 1 liter (l) air when that liter of air of can hold at most $58mg$, the humidity is

$$\frac{34mgl^{-1}}{58mgl^{-1}} = .586 = 58.6\% \tag{16.4}$$

Another common example of a unitless quantity is a probability. Saying that there is a 30% chance of rain means that under similar circumstances that rain would be expected 30 days out of 100, *i.e.* $30days/100days = .3 = 30\%$. One way to represent a probability is using a so-called Venn diagram, where a box represents the universe of possible events, and the area of subsets within that box represent the relative probability of that a given event will occur. For example, figure 16.2 is a pictorial representation that there is a 30% chance of rain. (See below for more on the interpretation of probabilities.)

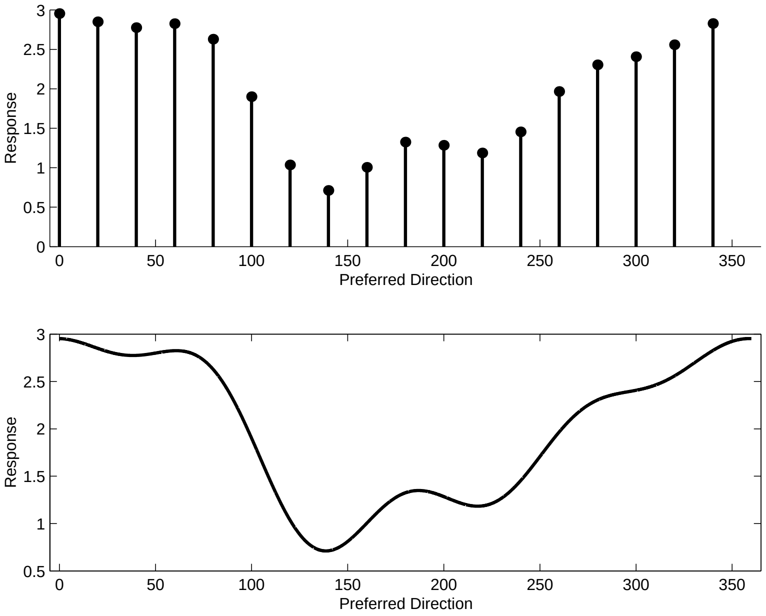


Figure 16.2: Venn Diagram Depicting a 30% chance of rain.

Sometimes whether a quantity has units is simply a matter of convention. For example, angles are often expressed in units of degrees (x^o). However, trigonometric functions require angles that are expressed in units of radians. Like percentages, radians denote a particular kind of ratio, namely the ratio of the length (l) around the circular tracing out the given angle over the radius (r) of the circle (figure 16.3).

In addition to appropriately rescaling fixed quantities, keeping track of units can be quite useful for understanding and interpreting an equation. For example, let's consider a very simple model of

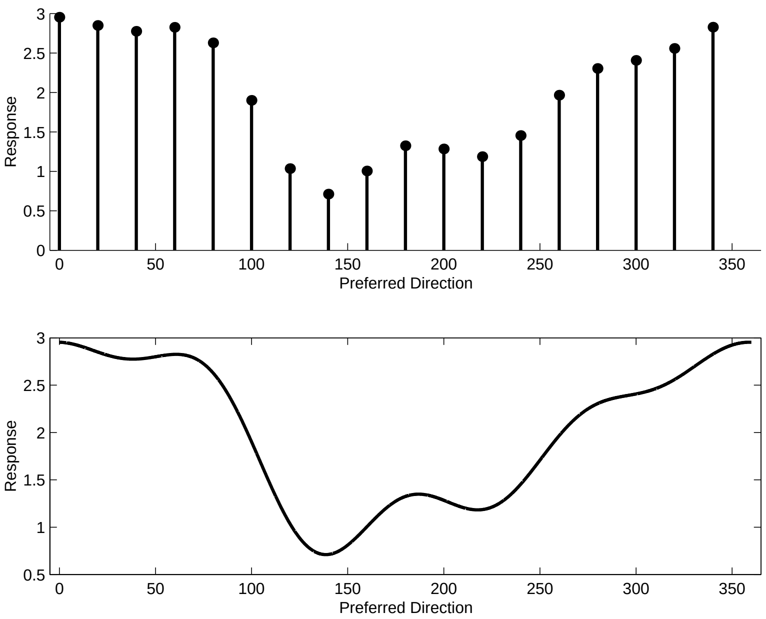


Figure 16.3: Angle in degrees and radians

neural firing where the firing rate is proportional to the average level of membrane potential above some threshold (this is explained in further detail in chapter 5):

$$\mathbf{r} = g(V - V_{\text{thresh}}) \tag{16.5}$$

Here, $\mathbf{r}$ represents firing rate and has units of $spikes/sec = Hz$, and V is average membrane voltage and is given in millivolts (mV). V_{thresh} is the threshold for spiking. What units should it have? From looking at the equation we see that we need to subtract the threshold value from the voltage. When adding or subtracting quantities, they must be in the same units (you can't add - or subtract for that matter - apples and oranges). So the threshold V_{thresh} should be also expressed in mV. Of course this makes sense: the voltage has to be above a certain number of millivolts for the neuron to begin spiking. g is the gain parameter that gives the proportionality between rates and voltage. What units should it have. Just looking at equation (16.5), g should have units of Hz/mV. That way, when it multiplies the voltage term $(V - V_{\text{thresh}})$ the units of mV cancel out, leaving the right side to have units of Hz. Then the quantities on both sides of the equation will be expressing the same thing, namely a firing rate.

One thing that should be bothering you about equation (16.5) is that when the membrane voltage is below threshold ($V < V_{\text{thresh}}$), the firing rate would be negative. One way to fix this is to write

$$\mathbf{r} = g\left[V - V_{\text{thresh}}\right]^{+} \tag{16.6}$$

where $[x]^{+}$ is equal to the maximum of x and 0 (is equal to x, if x is positive and is equal to 0 otherwise).

Problems

For the following problems you may need to consult additional reference material to convert some of these quantities.

Problem 16.2.1 Suppose an SUV gets 10 miles per gallon. How many kilometers per liter does it get?

MORE probs...

Problem 16.2.2 Convert 75 degrees Fahrenheit (^oF) to the Celsius scale (^oC). Why is this conversion different than other conversions done above?

16.3 Calculus

Understanding how to go to the continuum limit is what calculus is all about. Thus the fundamental contribution of calculus is to clearly define the notion of limit. The field of calculus then goes on to focus on two particular kinds of limits: the derivative and the integral. Given calculus' focus on these two limits, it is not surprising that the theorem describing the relationship between the derivative and the integral is known as the **fundamental theorem of calculus**.

16.3.1 The Derivative

To make things concrete, let's consider the velocity v of a given visual stimulus as a function of time t. We write $v(t)$. Suppose we want to know how fast the velocity is changing at a given time t, *i.e.* we want to know the acceleration at t. An approximation to the acceleration can be obtained by finding the difference between the velocity $v(t)$ at time t, and the velocity $v(t+dt)$ at some slightly later time $t+dt$, and then dividing by the time difference dt (figure **??**A). The derivative dv/dt or $v'(t)$ is what you get from the limit of this process as you use shorter intervals dt. Graphically, the derivative $v'(t)$ is the slope of the tangent line to the graph of the function $v(t)$ at time t.

Notational Aside. Derivatives with respect to time are very common, so sometimes a special notation $\dot{v} = dv/dt$ is used to represent time derivatives.

16.3.2 The Integral

Now suppose that we want to find out how far the stimulus has moved between time t_1 and t_2. Graphically, this is just equal to the area under the function $v(t)$ between t_1 and t_2 (figure **??**B). Conceptually, the dependent variable, time, is discretized by breaking up the t axis into small chunks or "bins" of length dt. For each bin indexed by i, we find the velocity $v(t_i)$ for some time t_i within that bin and calculate the distance travelled as that velocity times the width of the bin ($dtv(t_i)$, proportional to the area of a thin rectangle covering the bin of the appropriate height - figure **??**B). The total distance D is found by adding up all the rectangles:

$$D = \sum_{1}^{N} dt\ v(t_i) \tag{16.7}$$

Reducing the width of the time bins gives a more and more accurate estimate of the true area under the curve. The limit of this product is just the integral

$$D = \int_{t_1}^{t_2} dt\ v(t) \tag{16.8}$$

In many problems, it quite useful to keep track of the physical units of the quantities being represented. If what you are doing is right, the units have to work out there way through the calculations so that quantities on both sides of an equation represent the same kind of thing. For example, velocity might be expressed in meters per sec (m/sec). The derivative of velocity dv/dt comes from dividing by a time quantity so it is expressed in m/sec^2. In calculating the integral, we multiplied by the time quantity dt. We obtain something in the units $sec\ xm/sec = m$, this checks out with our intent that the integral should represent a distance.

16.3.3 Convolutions

16.3.4 Series Expansions

16.3.5 Exponential Decay

Suppose you start off with a fixed amount of radioactive material, say z kilograms, and a small fraction α (alpha) decays every second. How much material are you left with at time t? The answer is $ze^{-\alpha t}$, *i.e.* the material decays exponentially. Since we'll encounter versions of this problem over and over again, let's take it apart a bit. MORE.

16.4 Probability

Brain activity is notoriously variable. For example, sensory neurons respond somewhat differently, even when an animal is presented with the exact same stimulus. Thus most statements made about brain function are statistical, and to understand statistics one needs to understand the basic tenets of probability.

16.4.1 Definition of Probability

When you flip a coin, we say that there is a probability of 1/2 that the coin will end up heads. What do we actually mean when we say this? How do we know the probability is 1/2? There are two basic philosophical strategies put probability theory on a rigorous footing. Under the "**objective probability**" or "frequentist" interpretation, probabilities represent relative frequencies. So saying that the probability of a head is 1/2 simply means that if you flipped the coin 1000 times, on average you'd get 500 heads. While this statement may be true, it has some difficulties as a *definition* of probability. Suppose one flipped a coin that was bent so that it had a greater chance of ending up a head than a tails. How would you figure out the probability of a head? If you flipped it 1000 times and got 600 heads, you could say that the probability of a head was 600/1000 = 0.6. But suppose you repeated the experiment and this time got 590 heads. Is the probability of head 0.6 or 0.59? The way out of this dilemma is to define the probability of getting a head as the relative frequency of heads *in the limit of a large number of flips, i.e.* if the "true" probability is 0.6, then the relative frequency should get closer and closer to 0.6 as the number of flips grows very large.

Another problem with relative frequency as a definition is that it cannot be applied to unique events. When we flip a coin many times, we assume that we are performing a repeated series of identical "experiments." But what about the statement made before entering a labor negotiation that there is a high probability of a settlement? Presumably this statement rests on a combination of factors that are unique to that situation and so can't be viewed as a relative frequency.

To avoid the difficulties of defining probability as relative frequency, one can also view probability as a "belief" or "likelihood" that an event will take place. Since different people can have different opinions about the likelihood of an event, this is called the **subjective probability** point of view. The main objection to the subjective point of view is that it seems very hard to base science on a definition of probability that varies from person to person. However, all schemes for defining subjective probability require that the assigned likelihoods follow basic laws of probability (*e.g.* one cannot simultaneously "believe" that the probability of heads is 75% and the probability of tails is also 75%) and also require that the estimated probability agree with the relative frequency in the limit of unlimited amounts of data. As an example of how subjective probability might work, suppose that someone gave you a coin and you flipped it 10 times and got 9 heads. You would probably think that this is an unlikely event, but still assume that the probability of a head is 1/2. But if you flipped the coin 1000 times and got 900 heads, you'd probably think that something is funny with the coin. The reason that you might still think that 0.5 is the most likely probability for a head after 10 tosses is based on your prior belief (or *a priori* belief to use the Latin) that a normal looking coin should give equal probabilities for heads and tails. However, after 1000 flips your guess about the most likely probability of a head is likely to have changed. (This is known as an *a posteriori* or "after the fact" estimate.) Note here that when using subjective probabilities, you usually consider the range of possibilities of the underlying probability, and focus on the likelihood of which value is correct, *i.e.* you look at the likelihood of a likelihood. In the coin flip example, you started out thinking that 0.5 was the most likely likelihood of a head, but revised your estimate (maybe to 0.9) after trying 1000 flips. One of the main benefits of working with the subjective point of view is that calculations require you to formalize your prior expectations, and the method by which these are updated after getting some information.

The adherents of objective and subjective definitions for probability have been arguing back and forth in the statistics community for years. However, in the vast majority of situations, it is pretty clear what is meant by probability and it is fine to think of probability simply as some measure of likelihood, regardless of whether this is viewed as a relative frequency or subjective belief.

16.4.2 Properties of Probability

To introduce some of the basic properties of probability, consider the statement that there is a 30% chance of rain tomorrow. To formalize this, we denote the event that there is rain tomorrow by the letter A, and we write $P(A) = 0.3$. To view $P(B)$ as the likelihood of event B, we must have $0 \leq P(B) \leq 1$.

16.4.3 Random Variables

16.4.4 Probabilities with Continuous Variables

Another field where it is easy to blur continuous and discrete approaches to problems is probability theory. Suppose we flip a biased coin where the probability of getting heads is twice as great as getting tails. Since probabilities must add up to 1, we find that the probability of getting head is 2/3 while the probability of getting tails is 1/3.

Now consider a highly artificial problem where a sound is generated at a random horizontal position along a 1 meter long track. What's the probability that the sound is presented 2/3 of the way (reading from left to right) along the track? Since there are an infinite number of positions between $0.0m$(the left end) and $1.0m$ (the right end), a little thought reveals that the probability of choosing *exactly* the value $.6666\ldots m$ is infinitely small. The probability that the sound is between $.6m$ and $.7m$ should be 1/10, the probability that the sound is between $.66m$ and $.67m$ should be 1/100, etc. The probability that it falls exactly on a point should be less than the length of any interval that encompasses that point. Since such intervals can be arbitrarily small, the probability has to be equal to 0. So how can we talk about probabilities for this continuous problem? For example, suppose that the sound is generated twice as often on the left half of the track as on the right half. The way to solve this problem is to take our intuitions about probabilities over intervals, and to construct a **probability density function**, $\rho(x)$ that captures the relative probabilities of values characterized by the location x. such that the probability of finding a value in a small interval $[x, x+dx]$ is approximately equal to $dx\rho(x)$. Going now to the continuous limit, what we really want is that for any interval $[x_1, x_2]$, we want the probability that a sound falls within this interval to be given by the integral

$$\int_{x_1}^{x_2} dx\ \rho(x) \tag{16.9}$$

Of course the total probability must integrate to 1, *i.e.* in our problem

$$\int_0^1 dx\ \rho(x) \tag{16.10}$$

Given these constraints, the probability density function that is appropriate for our problem is

$$\begin{aligned}\rho(x) &= 4/3, \qquad (0 < x < .5) &(16.11)\\ &= 2/3, \qquad (.5 < x < 1) &(16.12)\end{aligned}$$

What units does $\rho(x)$ have? First, we need to remember that probabilities are pure (unitless) numbers. More generally, *any* value that is expressed as a ratio between two like quantities is unitless, since the units divide out. Probabilities can be thought of as the number of times that a certain event happens, divided by the total number of events. Returning to $\rho(x)$, recall that it was defined by the property that integrating with respect to x gives a probability. *Therefore, the units of the probability density $\rho(x)$ are be 1 over the units of x.* In our problem, $\rho(x)$ has units m^{-1}.

16.4.5 The Binomial Distribution

16.4.6 The Poisson Distribution

16.4.7 The Gaussian Distribution

16.4.8 The Central Limit Theorem

16.5 Complex Numbers

Complex numbers come up in a range of applications, and it will be particularly convenient to have them around when analyzing oscillatory phenomena. Complex numbers are numbers that can be written in the form $z = a + bi$, where a and b are ordinary real numbers and i is defined to be equal to $\sqrt{-1}$. (Note that while i is most commonly used to represent $\sqrt{-1}$, engineers tend to use the letter j.) a is known as the **real part** of the complex number z; bi is the **imaginary part**. Most mathematical operations that you can perform on real numbers can be performed on complex numbers. The definitions for addition and subtraction are trivial:

$$\begin{aligned}(a+bi)+(c+di) &= (a+c)+(b+d)i &(16.13)\\ (a+bi)-(c+di) &= (a-c)+(b-d)i &(16.14)\end{aligned}$$

Multiplication works out just like you'd expect:

$$(a+bi)(c+di) = ac + bdi^2 + (ad+bc)i = ac - bd + (ad+bc)i \tag{16.15}$$

However, multiplication has a non-intuitive geometrical interpretation that is quite useful. The complex number $a + bi$ can be viewed as a point or **vector** in the **complex plane**, by moving a distance a along the horizontal or **real axis** (sometimes referred to more generally as the x axis or **abscissa**) and a distance b along the vertical or **imaginary axis** (sometimes referred to as the y axis or **ordinate**)(figure **??**). Complex numbers are often represented by drawing an arrow to the appropriate location starting from the **origin** (the point $0 + 0i$). Addition of two complex numbers corresponds to operation of putting the two arrows tip-to-tail (figure **??**A). Subtraction corresponds to following the second arrow "backwards."

To understand multiplication, it is first useful to view the vector $a + bi$ in **polar coordinates**, *i.e.* by looking at its length r and direction θ. Recalling a bit of trigonometry,

$$\begin{aligned} r &= \sqrt{a^2 + b^2} && (16.16) \\ \theta &= \tan^{-1}(b/a) && (16.17) \\ a &= r\cos(\theta) && (16.18) \\ b &= r\sin(\theta) && (16.19) \end{aligned}$$

For fairly deep mathematical reasons the 17th century mathematician Leonhard Euler (XXXX-YYYY) first demonstrated that the appropriate formula for representing complex numbers in polar form is $re^{i\theta}$. Multiplying two complex numbers is pretty straightforward:

$$(r_1 e^{i\theta_1})(r_2 e^{i\theta_2}) = r_1 r_2 e^{i(\theta_1 + \theta_2)} \qquad (16.20)$$

It follows that to multiply two complex numbers, one multiplies their lengths and adds their angles (figure **??**B). This can be confirmed using equation (**??**) and the correspondences in equations (16.16)-(16.19) (see problem 16.5.1). This means that multiplication by $e^{i\theta}$ is equivalent to a counter-clockwise rotation through the angle θ. If we let t be time, then multiplying by $e^{2\pi i\omega t}$ causes a rotation whose angle continually increases with time. Since 2π represents one complete revolution, $2\pi\omega$ represents the number of cycles traversed per unit time.

16.5.1 The Complex Conjugate

Problems

Problem 16.5.1 Show that if (r_1, θ_1) and (r_2, θ_2) are the polar coordinate representations for the complex numbers $a_1 + b_1 i$ and $a_2 + b_2 i$, then $(r_1 r_2, \theta_1 + \theta_2)$ is the polar coordinate representation of $(a_1 + b_1 i)(a_2 + b_2 i)$.

Bibliography

Abeles M. (1991). *Corticonics: Neural circuits of the cerebral cortex.* Cambridge University Press.

Adrian E.D. (1964). *The Basis of Sensation.* Hafner, New York.

Albrecht D. G. (1995). Visual cortex neurons in monkey and cat: Effect of contrast on the spatial and temporal phase transfer functions. *Vis. Neurosci.*, 12:1191–1210.

Anderson J.A. (2000). *An Introduction to Neural Networks.* MIT Press, Cambridge, MA.

Batschelet E. (1979). *Introduction to Mathematics for Life Scientists.* Springer, New York, 3rd edition.

Brown T.H., Kairiss E.W., and Keenan C.L. (1990). Hebbian synapses: Biophysical mechanisms and algorithms. *Annu. Rev. Neurosci.*, 13:475–511.

Calvin W.H. and Stevens C.F. (1968). Synaptic noise and other sources of randomness in motoneuron interspike intervals. *J. Neurophysiol.*, 31:574–587.

Churland P.S. and Sejnowski T.J. (1992). *The Computational Brain.* MIT Press, Cambridge, MA.

Dayan P. and Abbott L.F. (2001). *Theoretical Neuroscience.* MIT Press, Cambridge, MA.

Frolov A.A. and Medvedev A.V. (1986). Substantiation of the 'point approximation' for describing the total electrical activity of the brain with the use of a simulation model. *Biophysics*, 31(5).

Gerstner W. (1999). *Spiking Neurons*, chapter 1, pages 3–53. MIT Press, Cambridge, MA.

Gerstner W. (2000). Population dynamics of spiking neurons: fast transients, asynchronous states, and locking. *Neural Comput.*, 12(1):43–89.

Hodgkin A. and Huxley A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (London)*, 117:500–544.

James W. (1983/1890). *The Principles of Psychology.* Harvard UP. Original work published in 1890.

Kandel E.R., Schwartz J.H., and Jessel T.M. (2000). *Principles of Neural Science.* Elsevier, New York, 4th edition.

Knight B.W. (1972). Dynamics of encoding in a population of neurons. *J. Gen. Physiol.*, 59(6):734–66.

Knight B.W. (2000). Dynamics of encoding in neuron populations: some general mathematical features. *Neural Comput.*, 12(3):473–518.

Koch C. (1999). *Biophysics of Computation: Information Processing in Single Neurons.* Oxford University Press.

Marr D. (1982). *Vision.* W.H. Freeman, New York.

O'Reilly R.C. and Munakata Y. (2000). *Computational Explorations in Cognitive Neuroscience.* MIT Press, Cambridge, MA.

Rieke F., Warland D., de Ruyter van Steveninck R., and Bialek W. (1997). *Spikes.* MIT Press, Cambridge, MA.

Softky W.R. and Koch C. (1993). The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *J. Neurosci.*, 13(1):334–350.

Troyer T.W. and Miller K.D. (1997). Physiological gain leads to high ISI variability in a simple model of a cortical regular spiking cell. *Neural Comput.*, 9(5):971–983.

Wilson H.R. and Cowan J.D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophys. J.*, 12(1):1–24.

Wilson H.R. and Cowan J.D. (1973). A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik*, 13(2):55–80.